PROJEST PROPOSAL For “Big Data Technologies and Applications VIA 504E - CRN: 21757”

Air Quality Data Platform

Harnessing Big Data Technologies In Favor Of More Sustainable Ways Of Living

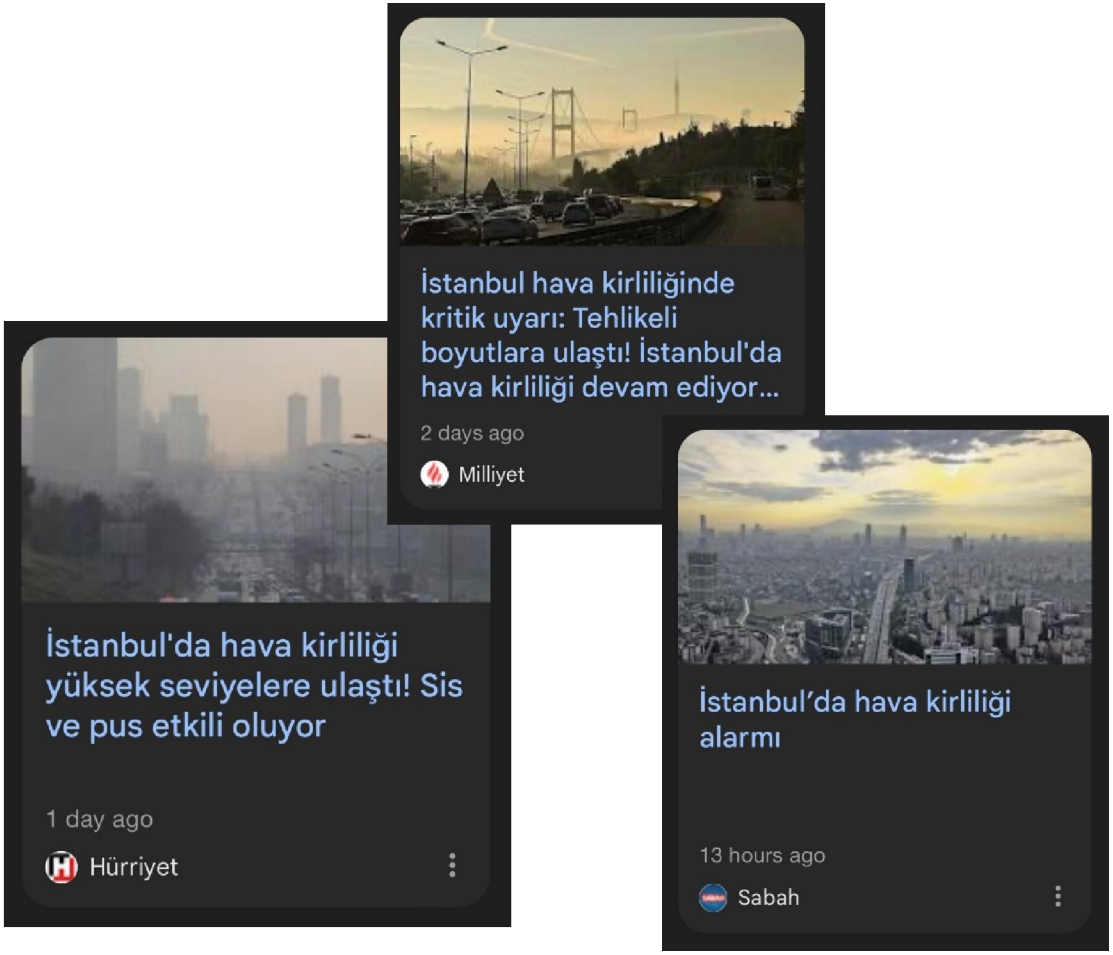
PRESENTED TO YOU BY

HORIZON

The Air We Can’t Escape

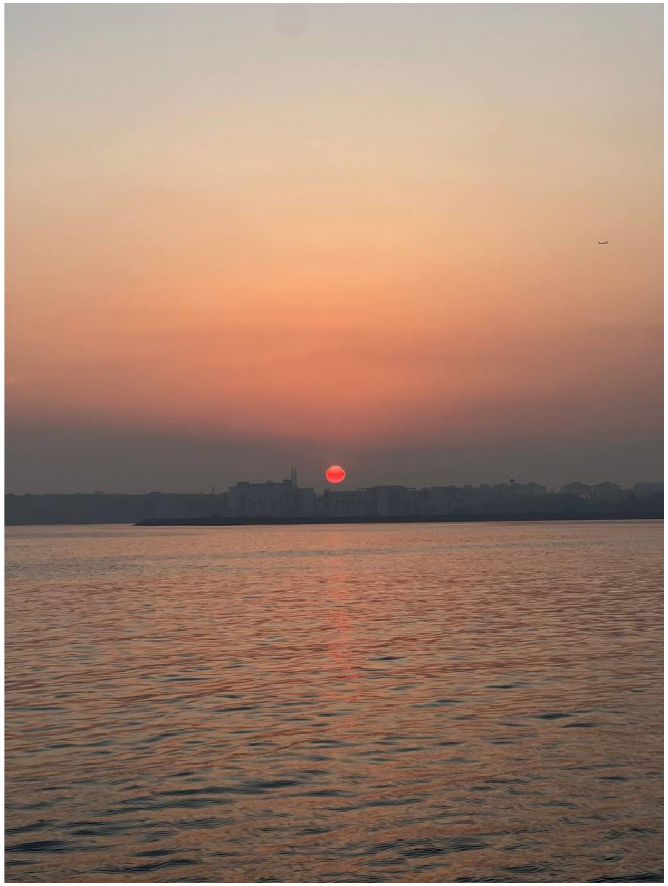
On the morning of February 1st, 2025, around the times when a group of teammates enrolled in Istanbul Technical University's Big Data program were overwhelmed by uncertainty about the topic for their term project, A team member woke up coughing and struggling to breathe, their house filled with the thick, heavy air of Istanbul’s traffic. Living by the congested E-5 highway, Pollution was a familiar issue, but that day, it felt noticeably worse.

A quick scroll through the news confirmed the suspicion—air pollution levels in Istanbul had surged to dangerous levels.



Headlines from Turkey's most trusted News sources on February 1st, 2025

From Awareness to Action



Baruthane Millet Bahçesi, February 1st, 2025

In search of fresh air and a peaceful sunset, the sea seemed like an escape. But instead, the view was swallowed by pollution—a darkened sky, a sun barely shining through the thick haze and the air unbreathable.

The camera captured it all, an image that could belong in a dystopian movie, yet it was the present. What started as a simple search for a project topic turned into purpose.

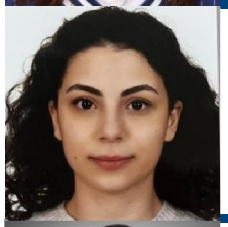
Joining Data-Driven Initiatives in the Sustainability Domain

HORIZON, a team committing to applying data-driven analysis to understand air pollution and its impact. Using the skills and knowledge gained, the goal is to contribute to sustainability efforts, ensuring that future horizons remain clear, and sunsets can be seen in their true colors once again.

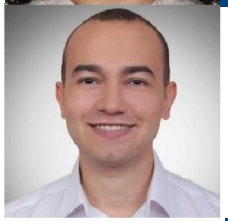
İSTANBUL TEKNİK ÜNİVERSİTESİ

HORIZON Team Members And Assigned Roles

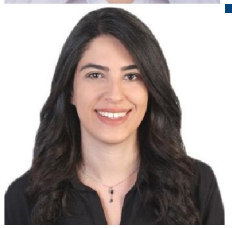
Project Manager: Aysan Pakmanesh
Student ID: 528241005



Domain Expert (Chemical Engineer): Melis Ciger
Student ID: 528241017



Developer: Samet Erkek
Student ID: 528241041



Reasercher: Selin Bozkurt
Student ID: 528241003

Why air quality?

Accroding to OpenAQ, Air pollution is the 2nd leading risk factor for death and the leading environmental contributor to death worldwide, contributing to more than 8 million deaths in 2021 [1]. Up to 99% of us are breathing polluted air [2], cutting lives short by an average of 1.9 years across the globe [3].

Air pollution impacts our health in many ways, starting when we're in the womb. Air pollution impacts practically every organ in the body and is associated with acute and chronic cardiovasculatory and respiratory illnesses, as well as increased incidence of lung cancer, diabetes, and poor birth outcomes. [4]
The burden of air pollution is unequal, with some people being far more vulnerable and some breathing far more polluted air. For example, children are particularly susceptible to air pollution's insidious effects—they breathe more rapidly than adults, taking in more polluted air, and are proportionally exposed at higher levels due to their small bodies and developing organs. Every single day, around 2,000 children under the age of 5 die from breathing polluted air. [5]

Marginalized communities are also unfairly burdened. Not only is the air they breathe typically more polluted, but they are also more susceptible to air pollution due to experiencing other poor health conditions caused by the broader social and environmental context in which they live. Structural racism and the legacy of colonialism play a large role in creating these health conditions and allowing for higher levels of exposure.

'Türkiye'de hava kirliliği, Kovid-19'dan daha fazla can aldı'

23:31 05.06.2021 (güncellendi: 23:40 05.06.2021)

https://anlatilaninotesi.com.tr/20210605/turkiyede-hava-kirliligi-kovid-19dan-daha-fazla-can-aldi-1044666780.html

Addressing air pollution has many benefits above and beyond better health.

- **A more stable climate:** Several pollutants both harm health directly and warm the planet.
- **Improved food security**: Cleaner air improves crop yields.
- **Healthier ecosystems, plants, and animals**: Air pollutants contaminate not only air, but also water and soil.
- **Greater social equity**: Reducing air pollution has even greater positive impacts for people who are disadvantaged socioeconomically and/or are disproportionately exposed.
- **Stronger economies**: Cleaner air results in a healthier, longer-lived workforce.
- **Clearer views**: Smog reduces visibility.
- **Preservation of culturally significant buildings and monuments**: Certain air pollutants form acid rain that not only harms living beings, but also degrades certain building materials.

3/11/25, 10:25 PM Türkiye'de hava kirliliğinin ekonomik maliyeti 1,5 trilyon lirayı aştı - Cumhur Haber Ajansı

Türkiye'de hava kirliliğinin ekonomik maliyeti 1,5 trilyon lirayı aştı

https://cumha.com.tr/turkiyede-hava-kirliliginin-ekonomik-maliyeti-15-trilyon-lirayi-asti

Air Quality Data

Despite the urgency of confronting this global challenge, only 61% of governments worldwide produce air quality data, leaving over 1 billion people without access to fundamental information that could protect them from the harmful effects of air pollution.[6]

Reliable data on air pollution are fundamental to understanding and taking corrective action to improve air quality. Air quality data underpin all actions on air pollution. They tell us how much pollution is in the air we breathe. They predict how air pollution will change in space and time. They shape our understanding of how air pollution impacts human health, climate change, ecological health and economic well-being.

How This Project Contributes to Air Quality Monitoring

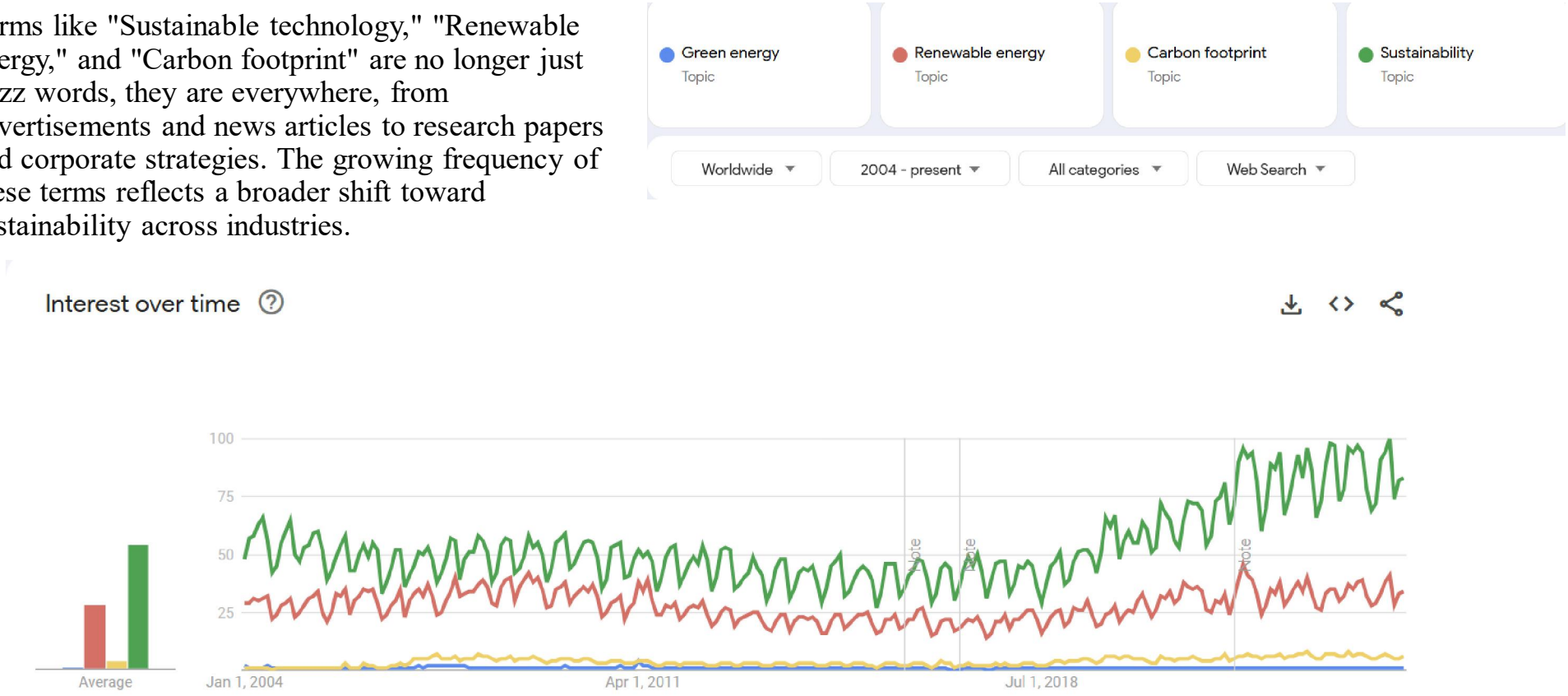
In this project, we aim to use our resources to contribute to the ongoing efforts of many researchers, environmentalists, and policymakers in the field of air quality monitoring. By leveraging data-driven approaches, we seek to enhance the accessibility and usability of air pollution data, making it easier for communities to understand and respond to environmental challenges. We hope to apply our learnings in utilizing **data as a service**—gathering pollution information, analyzing trends, and extracting meaningful insights. Through this process, we aim to bridge the gap between raw data and actionable knowledge, allowing us to identify pollution patterns and their long-term effects on human health and ecosystems. Ultimately, our goal is to transform air quality data into a powerful tool for awareness and decision-making, ensuring that individuals, organizations, and policymakers have the information they need to take meaningful actions toward a cleaner and healthier planet.

References

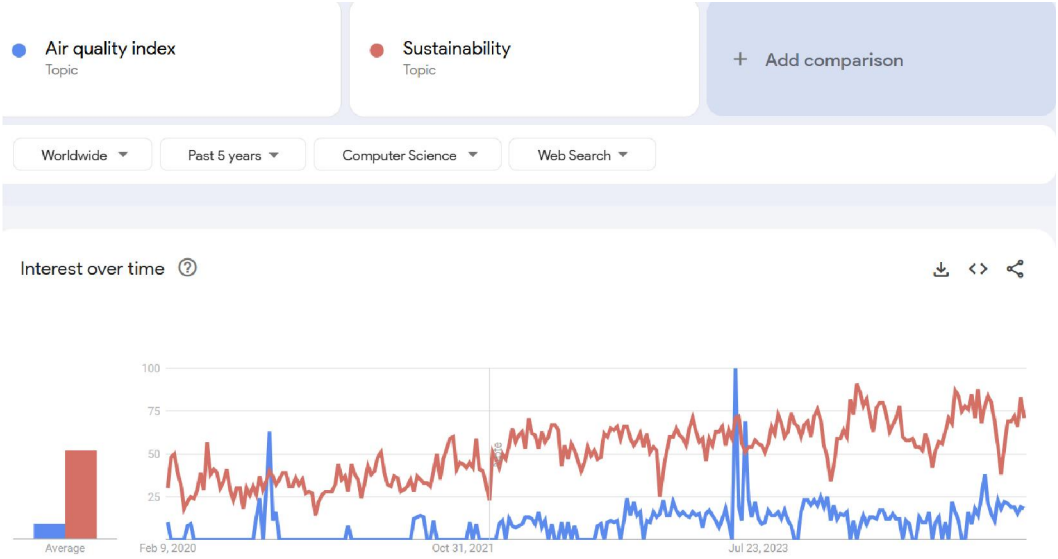
1. State of Global Air: Global Health Impacts of Air Pollution , Health Effects Institute and Institute for Health Metrics and Evaluation's Global Burden of Disease project ↩
2. World Health Organization: Air Pollution ↩
3. Air Quality Life Index 2024 , Energy Policy Institute at University of Chicago ↩
4. Health Impact of Air Pollution , American Lung Association ↩
5. Clean Air, Healthy Children , UNICEF ↩
6. Open Air Quality Data: The Global Landscape 2022 , OpenAQ ↩

Sustainable Technologies

Terms like "Sustainable technology," "Renewable energy," and "Carbon footprint" are no longer just buzz words, they are everywhere, from advertisements and news articles to research papers and corporate strategies. The growing frequency of these terms reflects a broader shift toward sustainability across industries.



However, a deeper analysis of trends in sustainability and air quality within computer science reveals a more significant insight: there is a rising demand for computation-related solutions in environmental monitoring and sustainable innovation. The increasing integration of AI, big data analytics, and IoT into sustainability efforts indicates that this field is becoming a necessity.



AI & Big Data Innovations in Sustainability

IoT for Smart Cities & Sustainable Urban Planning:

IoT-enabled sensors embedded in streetlights, traffic systems, and industrial zones provide real-time data on energy consumption, pollution levels, and waste management.

Example: Barcelona's IoT-powered smart waste management system reduced collection costs by 30% and improved recycling efficiency.

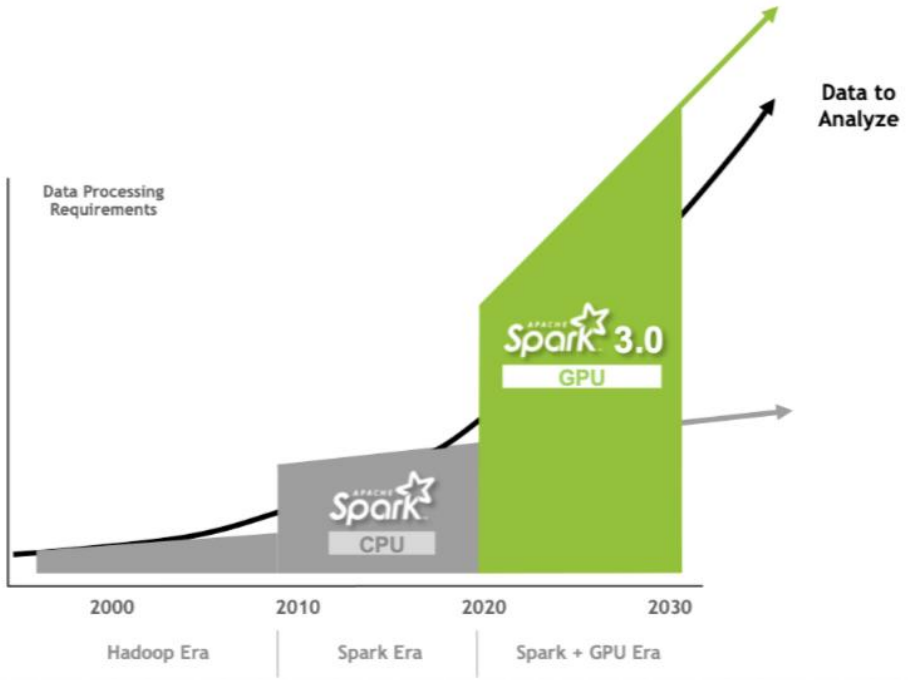
Big Data for Climate & Air Quality Monitoring:

Big data analytics processes millions of data points from IoT sensors, satellite imagery, and climate models to provide real-time monitoring of air quality, deforestation, and CO_2 emissions.

Example: NASA's Earth Observing System Data and Information System (EOSDIS) processes petabytes of satellite data to track deforestation and ocean temperature changes.

State Of The Art In Scaling Big Data Processing for AI and Sustainability:

Spark on NVIDIA GPU Clusters



https://developer.nvidia.com/blog/optimizing-and-improving-spark-3-0-performance-with-gpus/

As big data applications in sustainability grow more complex, the demand for faster, scalable, and efficient computing solutions has led to the integration of Apache Spark with NVIDIA GPU clusters.

Apache Spark, an open-source big data processing framework, is widely used for real-time analytics, machine learning, and large-scale data computations. When combined with NVIDIA's high-performance GPUs, Spark becomes exponentially faster, enabling massive-scale sustainability applications such as climate modeling, air quality predictions, and AI-driven environmental monitoring.

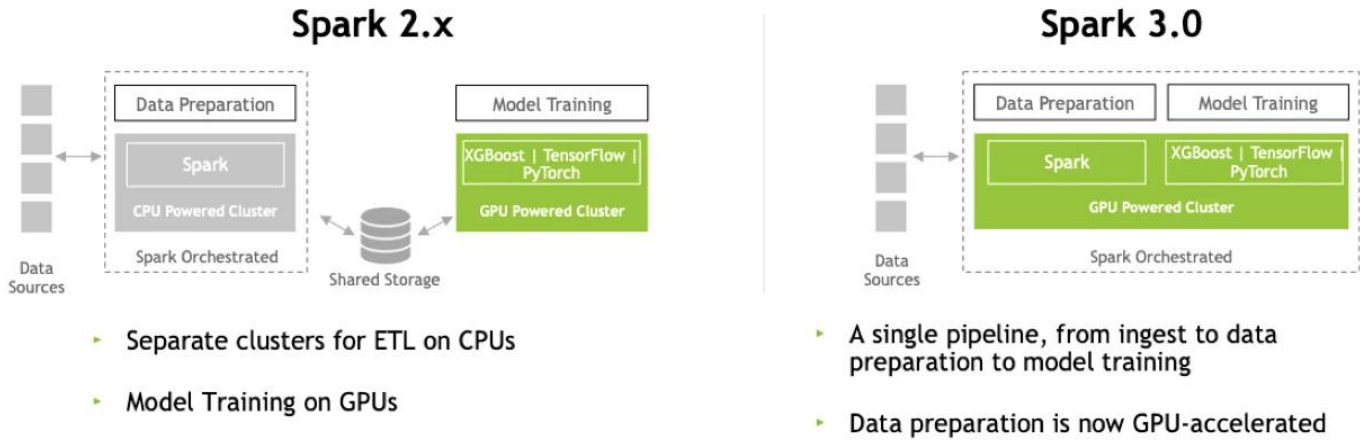
Key Features of Spark on NVIDIA GPUs:

Distributed Parallel Computing: Uses multiple GPUs across nodes to accelerate Spark jobs.

Memory Optimization: GPUs process massive datasets in-memory for real-time insights.

Scalability: Can scale up to thousands of GPUs across cloud infrastructures like AWS and Google Cloud.

RAPIDS GPU DataFrames: Provides GPU-accelerated Spark DataFrames, speeding up big data workflows 10x faster than CPU-based Spark.



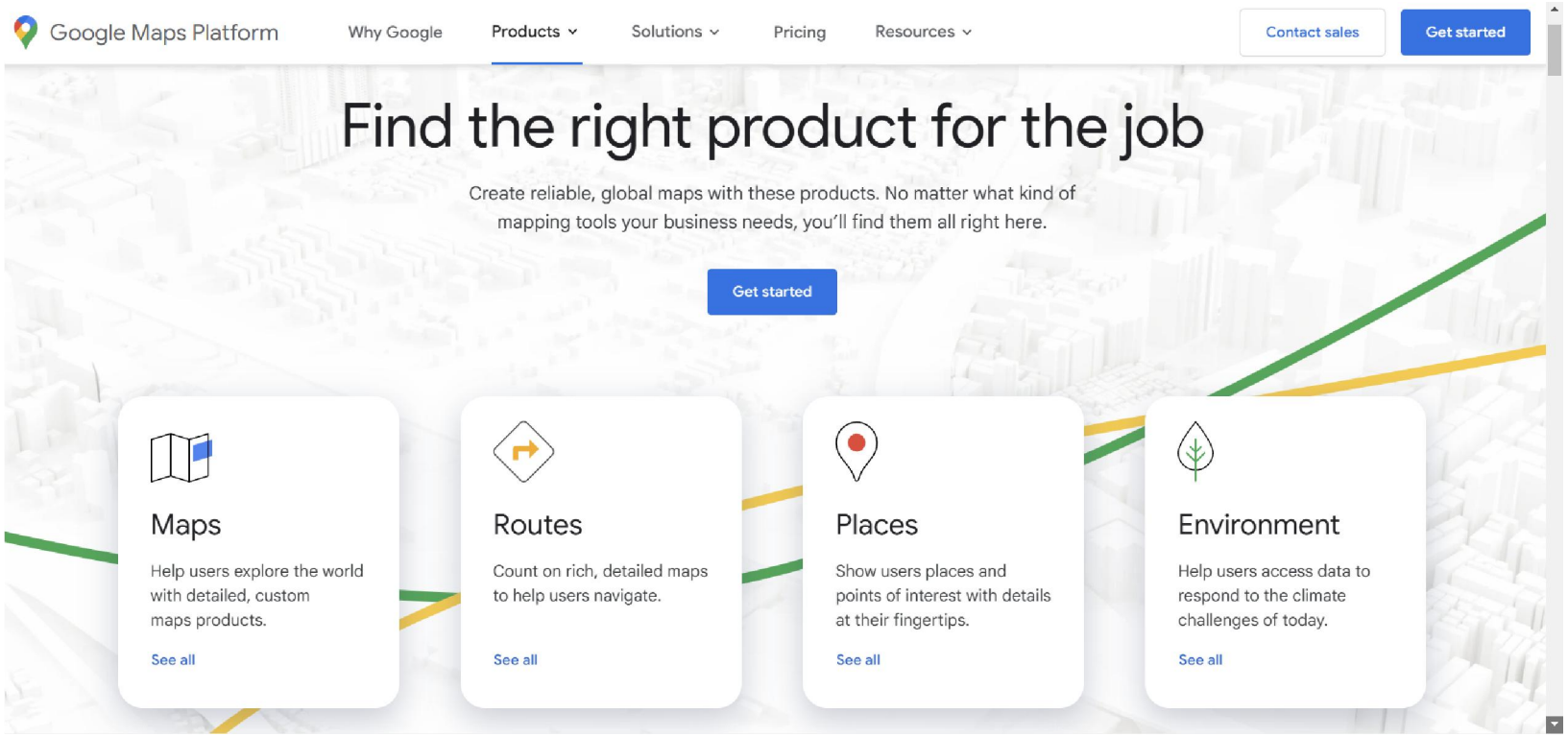
Website Credit: https://www.infoq.com/articles/deep-learning-apache-spark-nvidia-gpu/

In Spark 3.0, you can now have a single pipeline, from data ingestion to data preparation to model training on a GPU-powered cluster.

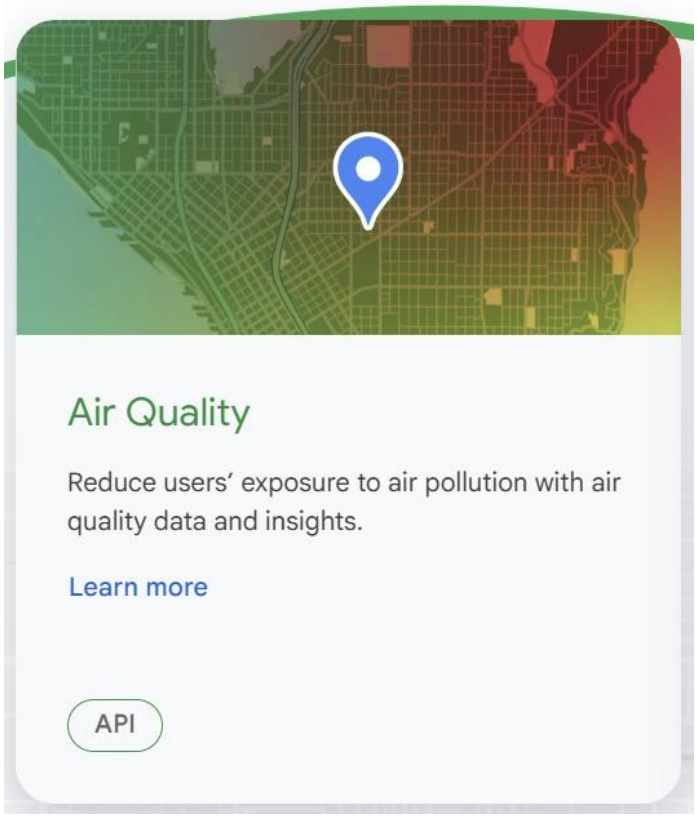
Example: Google's **Air Quality Index (AQI) monitoring** uses GPU-powered analytics to inform **traffic and pollution mitigation strategies**.

Business Vertical

Aim of our project is to build a cloud-based platform that collects, analyzes, and visualizes air quality data and even though we have initially began this journey with an open-end, it's still beneficial to discuss or imagine potentional business outcomes, Google Maps Platform is a good real life example use case scenario.



Google Maps Platform Overview



Google Maps Platform is a suite of APIs and SDKs that provide developers with advanced location-based services, enabling seamless integration of mapping, navigation, geospatial analysis, and real-time data into applications. It is categorized into four main areas:

Maps – Delivers interactive and customizable maps, satellite imagery, and Street View, allowing businesses and developers to create visually rich geospatial experiences.

Routes – Offers real-time traffic-aware routing, optimized navigation, and predictive travel time calculations, including eco-friendly travel options.

Places – Provides access to a vast location database, including business listings, geocoding, time zones, and local search capabilities, enhancing location intelligence and user experiences.

Environment – Focuses on sustainability-driven insights by offering data on air quality, solar potential, and climate impact, helping businesses and cities develop environmentally responsible solutions.

Sustainability Aims

Google Maps Platform plays a significant role in supporting sustainability by helping businesses and individuals make environmentally conscious decisions. The **eco-friendly routing feature** suggests fuel-efficient routes to reduce carbon emissions. The **Air Quality API** provides real-time pollution levels to inform users about outdoor conditions, while the **Solar API** helps assess the solar potential of rooftops to promote clean energy adoption. Additionally, urban planners, logistics companies, and governments use Google's environmental data to design more sustainable cities, optimize transportation networks, and mitigate environmental impact.

They also offer a variety of white papers on their website that explore their products in depth. One notable example is Driving Sustainability, a study conducted in partnership with Oxford Economics and Google, which examines how geospatial services contribute to sustainability across various industries.

Google, Driving Sustainability, 2023. [Online]. Available: https://mapsplatform.google.com/resources/whitepapers/?filterBy=inputIndustryAll

Sustainability is Becoming a Top Business Priority

Fig. 2: Organizational goals over the next three years

Google, Driving Sustainability, 2023. [Online]. Available: https://mapsplatform.google.com/resources/whitepapers/?filterBy=inputIndustryAll

As companies increasingly measure their environmental impact, the demand for data-driven sustainability solutions (such as geospatial analytics) is growing.

While organizations are still **primarily focused on operational efficiency (52%) and product/service quality (54%)**, sustainability is rapidly rising in importance. **38% of businesses** ranked sustainability as one of their **top three priorities**, ahead of **market share growth (31%) and speed-to-market (30%)**.

Fig. 1: Geospatial services reduce costs.

Despite progress, only **4% of companies have fully implemented sustainability initiatives**—leaving significant room for expansion.

Geospatial technology adoption is expected to rise, as businesses look for tools to track carbon emissions, optimize resource use, and improve environmental compliance.

What Is a Data Platform

A data platform is a comprehensive end-to-end solution that encompasses every part of the data management lifecycle. It allows a company to centralize its data solutions and enable efficient, autonomous management via one focal point; A central repository and processing house for all of an organization's data. A data platform handles the collection, cleansing, transformation, and application of data to generate business insights.

Website Credit For Graph: https://www.quantexa.com/resources/data-platform/

İSTANBUL TEKNİK ÜNİVERSİTESİ

Project Implementation Approach For
Air Quality Data Platform

Building a geospatial data platform requires a structured approach, integrating big data, AI-driven analytics, and cloud-based infrastructure to gather, process, and visualize environmental data.
Below is a step-by-step implementation plan, followed by challenges and solution ideas.

Initial Step: Data Gathering & Identifying the Right Source

The foundation of this project lies in accessing diverse, high-quality data sources to provide real-time and historical sustainability insights.

Implementation Challenge:
Many real-time datasets are scattered, inconsistent, or restricted in access.

Solution: Develop a data ingestion pipeline that aggregates multiple sources, cleans data, and ensures a uniform structure.

Data Structure	Sources & Websites	Big Data V's
Structured Data	- OpenAQ: openaq.org (Global Air Quality API, CSV exports) - EPA AirData: epa.gov/outdoor-air-quality-data (U.S. air quality data) - NASA EarthData: earthdata.nasa.gov (Global atmospheric data) - National Air Quality Monitoring Network (Turkey): havaizleme.gov.tr (Turkey's official air quality data)	Velocity: Regular updates, with some sources providing near real-time data. Variety: Different pollutants and measurement parameters across regions.
Semi-Structured Data	- AirVisual API: iqair.com/air-quality-api (JSON-based air quality data) - OpenWeather API: openweathermap.org/api (JSON/XML weather and air quality data) - Right to Clean Air Platform Turkey: temizhavahakki.org (Reports and data in various formats) - WAQI: *aqicn.org/api/*	Velocity: Real-time data streams from sensors and APIs. Variety: Data in JSON, XML, and other semi-structured formats.
Unstructured Data	- NASA Worldview: worldview.earthdata.nasa.gov (Satellite imagery) - Copernicus Atmosphere Monitoring Service: atmosphere.copernicus.eu (Global atmospheric composition data) - OpenALEX research data in PDF's and other formats - Research Reports: DergiPark (Turkish academic publications)	Volume: Large volumes of image and text data. Variety: Diverse data types requiring advanced processing techniques.

Step 2: Cloud-Based Data Storage & Processing

Once data is gathered, a scalable cloud storage solution is required to ingest, process, and manage large-scale datasets efficiently.

- **1. AWS S3: Scalable Data Lake for Storage**
- Amazon S3 (Simple Storage Service) serves as the primary data lake, handling structured, semi-structured, and unstructured sustainability data.

- **2. Apache NiFi: Real-Time Data Ingestion & Workflow Automation**
- NiFi (on AWS EC2) enables real-time data ingestion from multiple sources.

- **3. Apache Spark on AWS EMR: High-Performance Data Processing**
- Amazon EMR (Elastic MapReduce) is used to run Apache Spark for high-speed, distributed data processing.

Step 3: Geospatial Visualization & Business Intelligence (For Now, Will be Updated)

The final phase involves visualizing air quality & sustainability insights for businesses, policymakers, and consumers.

Scalability – Expanding as We Learn

This project is primarily focused on the **scale and complexity of data**, emphasizing how we store, process, and manage large datasets rather than diving deep into advanced machine learning models from the start. Since we are at the beginning of our **big data journey**, we are keeping our approach flexible, allowing us to adapt.

- **Data Volume Growth** → Over time, we can incorporate larger datasets from more sources
- **Variation in Data Types** → Expanding beyond traditional air quality metrics to include climate data, industrial emissions, and other environmental factors.
- **Machine Learning Progression** → Starting with basic trend analysis and gradually integrating more sophisticated AI models.
- **Geographical Expansion** → Beginning with a single city's air quality and eventually scaling to national and global levels.

Keeping it as **An Open-Ended Journey** As we progress, the direction of our expansion will be shaped by insights, challenges, and opportunities we encounter.

Examples of Similar Work

Air pollution monitoring has increasingly relied on big data analytics and machine learning (ML) to improve forecasting accuracy, enhance data collection, and generate real-time insights. The two studies analyzed here focus on big data handling techniques and ML-based predictive models for air quality analysis.

https://doi.org/10.1007/s11277-023-10351-1

The State-of-the-Art in Air Pollution Monitoring and Forecasting Systems using IoT, Big Data, and Machine Learning

Amisha Gangwar[1,2], Sudhakar Singh[1,*], Richa Mishra[1], Shiv Prakash[1]
[1] Department of Electronics and Communication, University of Allahabad, Prayagraj, India
[2] School of Information Studies, Syracuse University, New York, US

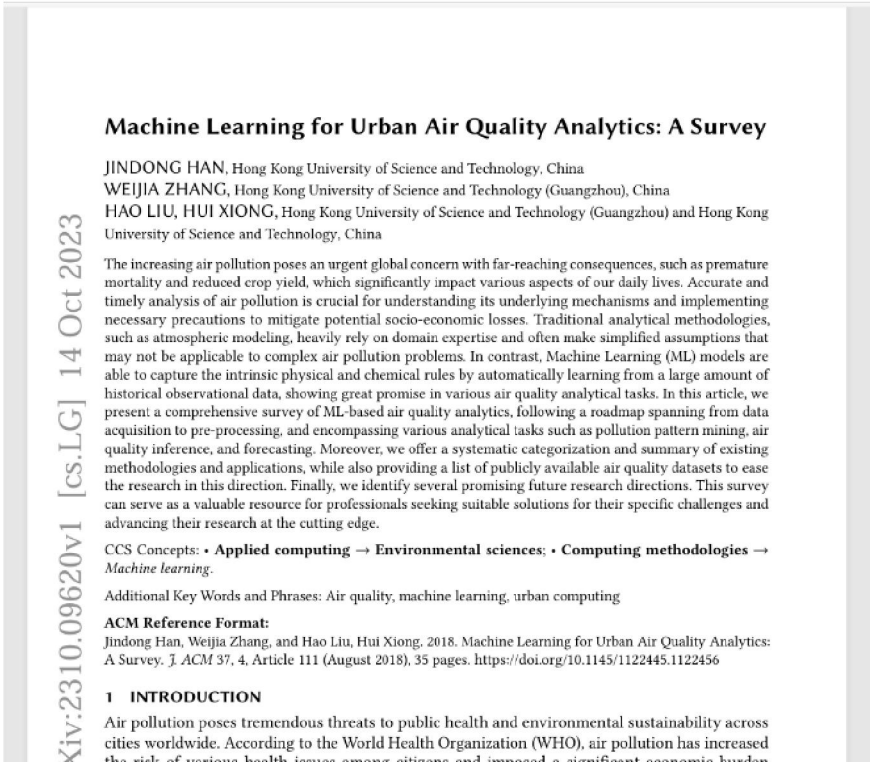
*Corresponding author
E-mail addresses:
agangwar@syr.edu (Amisha Gangwar)
sudhakar@allduniv.ac.in (Sudhakar Singh)
richa_mishra@allduniv.ac.in (Richa Mishra)
shivprakash.cse@gmail.com (Shiv Prakash)

Abstract

The quality of air is closely linked with the life quality of humans, plantations, and wildlife. It needs to be monitored and preserved continuously. Transportations, industries, construction sites, generators, fireworks, and waste burning have a major percentage in degrading the air quality. These sources are required to be used in a safe and controlled manner. Using traditional laboratory analysis or installing bulk and expensive models every few miles is no longer efficient. Smart devices are needed for collecting and analyzing air data. The quality of air depends on various factors, including location, traffic, and time. Recent researches are using machine learning algorithms, big data technologies, and the Internet of Things to propose a stable and efficient model for the stated purpose. This review paper focuses on studying and compiling recent research in this field and emphasizes the Data sources, Monitoring, and Forecasting models. The main objective of this paper is to provide the astuteness of the researches happening to improve the various aspects of air polluting models. Further, it casts light on the various research issues and challenges also.

Keywords: Air Pollution, Monitoring, Forecasting, Internet of Things (IoT), Big Data, Machine Learning (ML)

https://doi.org/10.1145/1122445.1122456



Machine Learning for Urban Air Quality Analytics: A Survey

JINDONG HAN, Hong Kong University of Science and Technology, China
WEIJIA ZHANG, Hong Kong University of Science and Technology (Guangzhou), China
HAO LIU, HUI XIONG, Hong Kong University of Science and Technology (Guangzhou) and Hong Kong University of Science and Technology, China

The increasing air pollution poses an urgent global concern with far-reaching consequences, such as premature mortality and reduced crop yield, which significantly impact various aspects of our daily lives. Accurate and timely analysis of air pollution is crucial for understanding its underlying mechanisms and implementing necessary precautions to mitigate potential socio-economic losses. Traditional analytical methodologies, such as atmospheric modeling, heavily rely on domain expertise and often make simplified assumptions that may not be applicable to complex air pollution problems. In contrast, Machine Learning (ML) models are able to capture the intrinsic physical and chemical rules by automatically learning from a large amount of historical observational data, showing great promise in various air quality analytical tasks. In this article, we present a comprehensive survey of ML-based air quality analytics, following a roadmap spanning from data acquisition to pre-processing, and encompassing various analytical tasks such as pollution pattern mining, air quality inference, and forecasting. Moreover, we offer a systematic categorization and summary of existing methodologies and applications, while also providing a list of publicly available air quality datasets to ease the research in this direction. Finally, we identify several promising future research directions. This survey can serve as a valuable resource for professionals seeking suitable solutions for their specific challenges and advancing their research at the cutting edge.

CCS Concepts: • **Applied computing** → **Environmental sciences**; • **Computing methodologies** → *Machine learning*.

Additional Key Words and Phrases: Air quality, machine learning, urban computing

ACM Reference Format:
Jindong Han, Weijia Zhang, and Hao Liu, Hui Xiong. 2018. Machine Learning for Urban Air Quality Analytics: A Survey. *J. ACM* 37, 4, Article 111 (August 2018), 35 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Air pollution poses tremendous threats to public health and environmental sustainability across cities worldwide. According to the World Health Organization (WHO), air pollution has increased

Challenges That They Faced & What Should Be The Next Move

While both studies focus on air quality analysis, they differ in many areas

Paper 1 is infrastructure-heavy (big data & IoT), Paper 2 is ML-centric (smart AI-driven analytics).

Aspect	Paper 1: IoT, Big Data, and ML	Paper 2: ML for Air Quality Analytics
Focus	IoT-based monitoring, big data processing, forecasting	ML-based analytics, pattern mining, inference
Big Data Handling	Uses Hadoop, Apache Spark, cloud computing for big data	Focuses on efficient ML models for high-dimensional data
Data Processing	Noise filtering, outlier detection, real-time data fusion	Sensor calibration, missing data imputation, data transformation
Machine Learning	Covers traditional ML models (SVM, Random Forest, etc.)	Covers advanced ML techniques (Deep Learning, Graph Neural Networks, Semi-Supervised Learning)
Forecasting Methods	Uses statistical and ensemble ML models	Uses spatio-temporal ML, autoencoders, attention-based models
Challenges	Scalability of storage and computing in IoT	Scalability of ML models for high-dimensional, spatio-temporal data
Applications	Smart cities, industrial monitoring, real-time systems	Scientific research, urban computing, ML model development

But these both studies have something in common and that the challenges they face, **they either handle big data well but lack predictive power for it or have strong ML models but struggle with real-world, large-scale data deployment.**

İSTANBUL TEKNİK ÜNİVERSİTESİ

Brief Summary of the Study: Geospatial Big Data Analytics for Sustainable Smart Cities

https://doi.org/10.5194/isprs-archives-XLVIII-4-W7-2023-141-2023

Objective of the Study

This study by M.O. Mete focuses on leveraging geospatial big data analytics to develop sustainable smart city solutions. The goal is to create an integrated technical infrastructure for Smart Environment and Smart Governance components within urban settings. The research specifically explores big data processing, parallel computing frameworks, and geospatial analytics to improve energy efficiency, pollution monitoring, and urban planning.

Key Findings

The Role of Geospatial Big Data in Smart Cities is increasing;

- As urbanization grows, cities generate massive geospatial datasets that are crucial for energy conservation, pollution control, and urban efficiency.
- Geospatial intelligence plays a vital role in tracking environmental changes, traffic congestion, air quality, and smart infrastructure management.
- Sustainable smart cities must rely on big data storage, analytics, and visualization tools to achieve green energy goals and carbon neutrality.

The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLVIII-4/W7-2023
FOSS4G (Free and Open Source Software for Geospatial) 2023 – Academic Track, 26 June–2 July 2023, Prizren, Kosovo

GEOSPATIAL BIG DATA ANALYTICS FOR SUSTAINABLE SMART CITIES

M. O. Mete [1] *

[1] Department of Geomatics Engineering, Istanbul Technical University, 34469 Istanbul, Türkiye - metemu@itu.edu.tr

KEY WORDS: Geospatial, Big Data, Parallel Computing, Smart City, Smart Environment, Smart Infrastructure, Green Energy

ABSTRACT:

Growing urbanization cause environmental problems such as vast amount of carbon emissions and pollution all over the world. Smart Infrastructure and Smart Environment are two significant components of the smart city paradigm that can create opportunities for ensuring energy conservation, preventing ecological degradation, and using renewable energy sources. Since a great portion of the data contains location information, geospatial intelligence is a key technology for sustainable smart cities. We need a holistic framework for the smart governance of cities by utilizing key technological drivers such as big data, Geographic Information Systems (GIS), cloud computing, Internet of Things (IoT). Geospatial Big Data applications offer predictive data science tools such as grid computing and parallel computing for efficient and fast processing to build a sustainable smart city ecosystem. Effective management of big data in storage, visualization, analytics, and analysis stages can foster green building, green energy, and net zero targets of countries. Parallel computing systems have the ability to scale up analysis on geospatial big data platforms which is key for ocean, atmosphere, land, and climate applications. In this study, it is aimed to create the necessary technical infrastructure for smart city applications with a holistic big data management approach. Thus, a smart city model framework is developed for Smart Environment and Smart Governance components and performance comparison of Dask-GeoPandas and Apache Sedona parallel processing systems are carried out. Apache Sedona performed better on the performance test during read, write, join and clustering operations.

1. INTRODUCTION

Increasing urbanisation across the world makes cities more crowded and complex. This situation brings along many social, environmental and economic problems such as housing, traffic density and air pollution. In order to overcome these problems and manage cities effectively, there is a need for sustainable smart cities that utilise information and communication technologies (ICTs) (Batty, 2013, Giffinger et al., 2007, Huang, Yao, Krisp, and Jiang, 2021).

Handling geospatial big data for sustainable smart cities is crucial since smart city services rely heavily on location-based data. Effective management of big data in storage, visualization, analytics, and analysis stages can foster green building, green energy, and net zero targets of countries. Geospatial data science ecosystem has many powerful open source software tools. According to the vision of PANGEO, a community of scientists and software developers working on big data software tools and customized environments, parallel computing systems have the ability to scale up analysis on geospatial big data platforms which is key for ocean, atmosphere, land, and climate applications. Those systems allow users to deploy clusters of compute nodes for big data processing.

A smart city can be defined as a modern city that utilises information technologies to improve its services and management and solve problems affecting the city (Berst, 2012, Li, Batty, and Goodchild, 2020). Emerging after 1990, this concept has brought about a paradigm shift by prioritizing replacing the purely digital city concept (Ateş and Erinsel Önder, 2019).

Computers, mobile phones, sensors, and even humans generate massive amounts of data, the size of which is increasing day by day (Batty, 2013). Geospatial big data has the potential to contribute to the development of smart cities, diagnosis of existing problems, prediction of changes in cities and optimised decision-making. On the other hand, intensive spatial data from various data sources paves the way for the development of many innovative applications related to cities. In the literature, there are different use cases of geospatial big data such as, social network analysis, mobility analysis and urban planning with communication network data (Calabrese, Ferrari, and Blondel, 2014, Dong, Wang, and Liu, 2021, Huang, Cheng, and Weibel, 2019), urban analysis and mobility with GPS data, event detection, sentiment analysis, travel orientation analysis and modelling of urban functions with location-based social media data (Hu, Mao, and McKenzie, 2018, Wei and Yao, 2021), transportation planning, intelligence and urban planning with smart transportation card data (Huang et al., 2021), shopping behaviour analysis, event management and building occupancy modelling with Wi-Fi and bluetooth data (Mashuk, Pinchin, Siebers, and Moore, 2021, Trasberg, Soundararaj, and Cheshire, 2021, Versichele et al., 2014), monitoring physical environments and human behaviour with camera images (Biljecki and Ito, 2021). The applications of the six basic components of smart cities contribute greatly to achieving the vision of sustainable smart cities.

Understanding Air Pollutants

When examining the air quality index, it is known that there is an important correlation between the meteorological elements concentration such as PM2.5, PM10, SO 2, NO2, O 3, CO, temperature and humidity.

PM2.5

PM2.5 refers to fine particulate matter in the atmosphere and having a diameter smaller than 2.5 micrometers.

Health Effects of PM2.5

1. Airway Damages

It has been observed that even small amounts of PM2.5 in the air people breathe might have serious effects on public health. Since the lungs are the first organ exposed to air when it is inhaled, there is a possibility that its toxic effects, which can cause airway inflammation, can be seen in the lungs.

2. Cardiovasculer Impairments

Several studies have found that PM2.5 can affect the autonomic nervous system (ANS), which controls the heart, and reduce heart rate variability (HRV). A low HRV is linked to a higher risk of heart disease and death.

3. Induction/exacerbation of diabetes mellitus

Studies have shown that long-term exposure to PM2.5 can contribute to diabetes by causing various problems related to type 2 diabetes (T2DM). These include insulin resistance (IR), visceral adipose inflammation, changes in brown adipose mitochondrial adipose, and stress in hepatic endoplasmic reticulum. Even low levels of PM2.5 exposure have been linked to a higher risk of death from diabetes.

4. Adverse effects in infancy

Even if it was intended that protecting the health of mothers and babies is as a global priority, but studies show that exposure to PM2.5 during pregnancy can cause problems like premature birth, low birth weight, and infant death. These issues not only affect newborns but can also lead to health problems in childhood and adulthood. Prenatal exposure to PM2.5 has been linked to a higher risk of lung infections and long-term breathing issues in young children. Since the immune system develops before birth, early exposure to air pollution can have lifelong effects on health.

According to the study conducted in 13 air sites at Xi'an University, PM2.5, PM10, SO2, NO2, O3, CO, temperature and humidity data were collected. The data was recorded by an automatic recorder for 24 hours and data was taken every hour. In the study, multiple linear regression analysis was performed for PM2.5 concentrations and a calculation model was created, and other pollutants in the air, PM10, SO2, NO2, O3, CO, temperature and humidity, were considered as variables that are correlated with PM2.5.

As a result of the study and calculations, a meaningful relationship was found between PM2.5 and other pollutants observed, as can be seen in the following results:

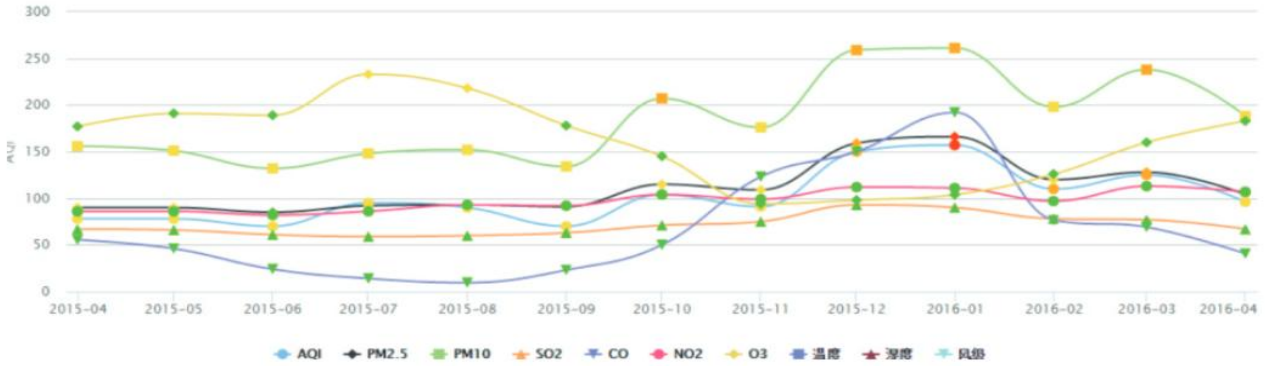


FIG. 1 The air quality graph of Xi'an (2015.4-2016.4)

Air Pollutants

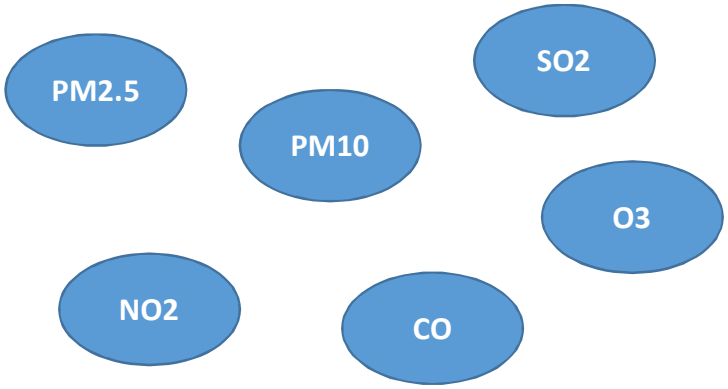


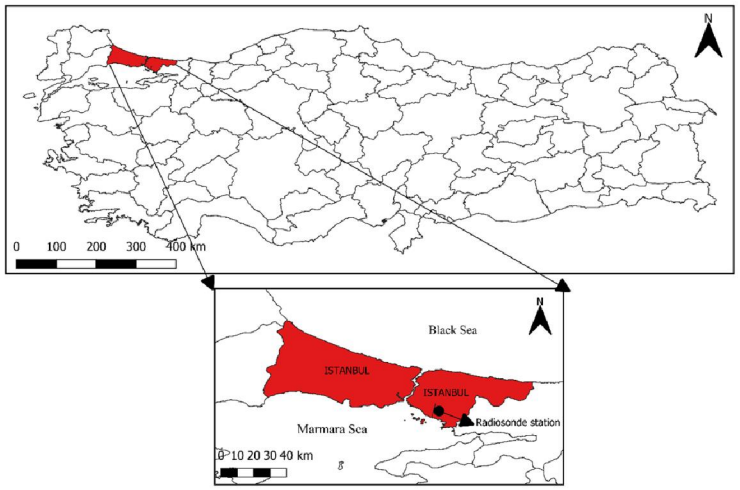
Table 2 Air pollutant threshold values

Air Pollutants	Duration	Limit values ($\mu g/m^3$)
PM_{10}	Daily	50
$PM_{2.5}$	Yearly	25
SO_2	Daily	125
NO_2	Hourly	200
CO	8-hour	10000
O_3	Hourly	240

Health Effects of Air Pollutants

- Pneumonia
- Heart rhythm disorders
- Diabetes
- Respiratory diseases

Major and Most Prevalent Air Pollutants



- PM10 and PM2.5 poses a serious risk to respiratory health in the city.
- Air pollution in Istanbul and the crowded cities in turkey is clearly associated with increases in hospital visits for common respiratory problems.
- Stable atmospheric conditions lead to higher pollution levels

Table 3 Monthly and annual threshold value exceeding rates of PM_{10} concentration values

PM_{10}	Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Basaksehir	38%	38%	47%	53%	55%	46%	27%	15%	15%	31%	36%	48%	41%
Esenyurt	66%	56%	58%	73%	74%	72%	72%	59%	59%	60%	63%	65%	63%
Kandilli	20%	27%	29%	35%	32%	19%	9%	3%	3%	8%	18%	33%	26%
Mecidiyekoy	49%	50%	55%	63%	61%	52%	42%	30%	28%	37%	46%	58%	47%
Silivri	16%	18%	21%	27%	32%	13%	6%	6%	4%	4%	14%	24%	16%
Sile	4%	3%	5%	7%	3%	5%	1%	1%	1%	2%	6%	6%	2%
Sirinevler	39%	42%	43%	57%	56%	44%	32%	16%	20%	30%	35%	48%	41%
Umraniye	35%	37%	37%	44%	44%	34%	22%	18%	24%	25%	36%	52%	40%
Uskudar	20%	27%	31%	34%	33%	20%	6%	2%	2%	5%	17%	32%	25%
Average	32%	33%	36%	44%	43%	34%	24%	17%	18%	22%	30%	41%	33%

Next Steps & Timeline

Week 1 (Feb 11 - Feb 18): Data Gathering
- Collect air quality data.
- Set up **AWS S3 and Apache NiFi**

Week 2 (Feb 19 - Feb 25): Data Cleaning & Preprocessing
- Process, clean, and normalize datasets.
- Implement **ETL pipelines** for structured storage.

Week 3 (Feb 26 - Mar 4): Implementation (Big Data Infrastructure)
- Deploy **AWS EMR, Spark** distributed processing.

Week 4 (Mar 5 - Mar 11): Visualization & Dashboard
- Getting insights and doing Analysis (On Gdelt Data as well)

Week 5 (Mar 26 - …): Final Touches
- Complete documentation, refine insights.

Presentation & Future Roadmap
- Deliver results, plan next steps.

This document outlines the steps taken to ingest and process large-scale datasets from public sources into our AWS S3 bucket. The implementation was designed to leverage EC2 instances with other provided tools, meanwhile handling the constraints of limited access to resources and keeping infrastructure as budget-conscious as possible, minimizing resource consumption.

Implementation Journey Documentation

The goal of this project is to efficiently collect, process, and store large-scale air quality-related data and after doing some testing we found three primary resources that were accessible and fit our needs best:

OpenAlex – A vast open dataset of scholarly works, where we extract research articles and PDFs related to air quality, sustainability, and environmental science, alongside technological research done for these field.

OpenAQ – A global open-source air quality data platform that provides historical air pollution measurements globally.

Hava İzleme – Turkey's national air monitoring system, which provides real-time and historical detailed air quality metrics for a vast number of locations.

1. OpenAlex

Overview of OpenAlex
OpenAlex is a comprehensive, open-access database that maps scholarly entities and their interconnections. It includes works (research papers), authors, journals, institutions, topics, publishers, and funders, forming a heterogeneous directed graph with hundreds of millions of entities and billions of relationships.

Users can access OpenAlex data through:

•**A free REST API** (limited to 100,000 requests per day, with optional premium services for higher access).

•**A complete database snapshot**, updated monthly and available for bulk downloads via AWS S3.

Snapshot data format

```
openalex-snapshot/
├── LICENSE.txt
├── RELEASE_NOTES.txt
└── data
    ├── authors
    │   ├── manifest
    │   └── updated_date=2021-12-28
    │       ├── 0000_part_00.gz
    │       └── 0001_part_00.gz
    ├── concepts
    │   ├── manifest
    │   └── updated_date=2021-12-28
    │       ├── 0000_part_00.gz
    │       └── 0001_part_00.gz
    ├── institutions
    │   ├── manifest
    │   └── updated_date=2021-12-28
    │       ├── 0000_part_00.gz
    │       └── 0001_part_00.gz
    ├── sources
    │   ├── manifest
    │   └── updated_date=2021-12-28
    │       ├── 0000_part_00.gz
    │       └── 0001_part_00.gz
    └── works
        ├── manifest
        └── updated_date=2021-12-28
            ├── 0000_part_00.gz
            └── 0001_part_00.gz
```

Initial Attempt to Download the Full OpenAlex Dataset

In theory, OpenAlex provides a full dataset snapshot hosted publicly on AWS S3 (s3://openalex/), accessible without authentication. This dataset can be retrieved using:

aws s3 sync "s3://openalex" "openalex-snapshot" --no-sign-request

Challenges Encountered

However, we quickly ran into significant issues with this approach:

1. Storage Limitations

•The complete OpenAlex dataset exceeds **1.5 TB** when unzipped, which surpassed the available storage capacity of our local computers and assigned EC2 instances by far, which only provided approximately **16 GB** of storage space at max. Upgrading to larger instances with sufficient storage was deemed impractical due to budget constraints.

2. Bucket-to-Bucket Transfer Issues

•Direct transfers between two public AWS S3 buckets (s3://openalex/ to s3://horizon-bigdataingestion/) were not possible due to AWS limitations on cross-bucket operations without intermediary storage or processing.

3. IAM Role and AWS Configuration Constraints

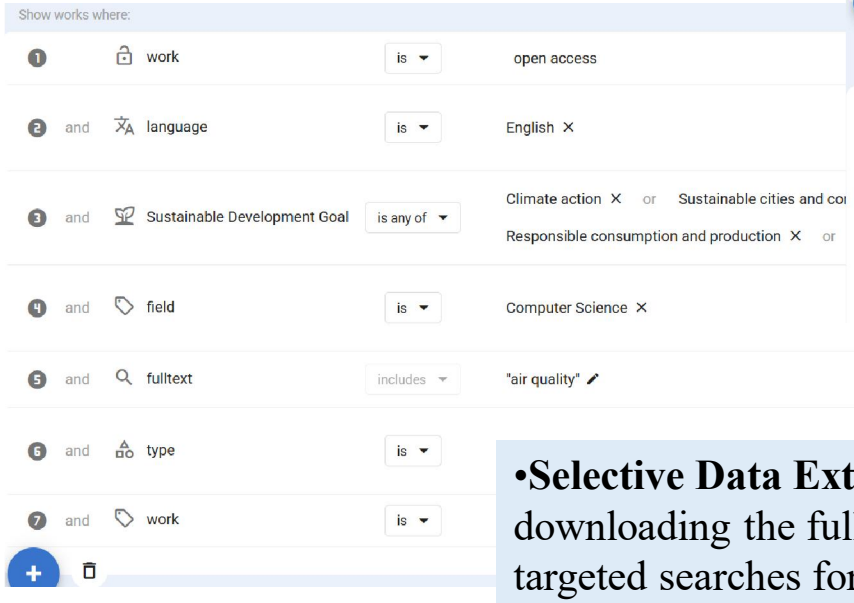
•Our project restrictions explicitly prohibited assigning IAM roles to EC2 instances. Attempts to use the standard AWS configuration (aws configure) on EC2 instances without IAM roles led to authentication and permission issues, rendering some instances inaccessible. This significantly complicated our workflow and halted progress at various stages.

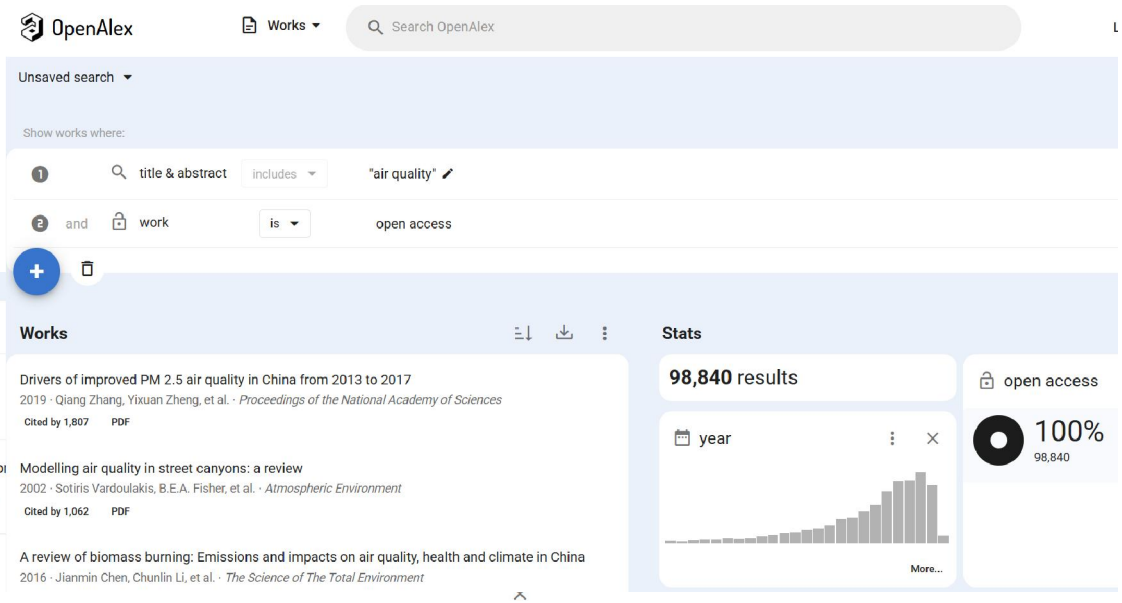
4. Inefficient Data Retrieval

•Downloading large datasets over the network with single-threaded approaches like using boto3 in a python script proved inefficient, resulting in inconsistent download speeds and prolonged retrieval times.

Alternative Strategy Implemented

Given these limitations, our team devised an alternative, optimized solution:

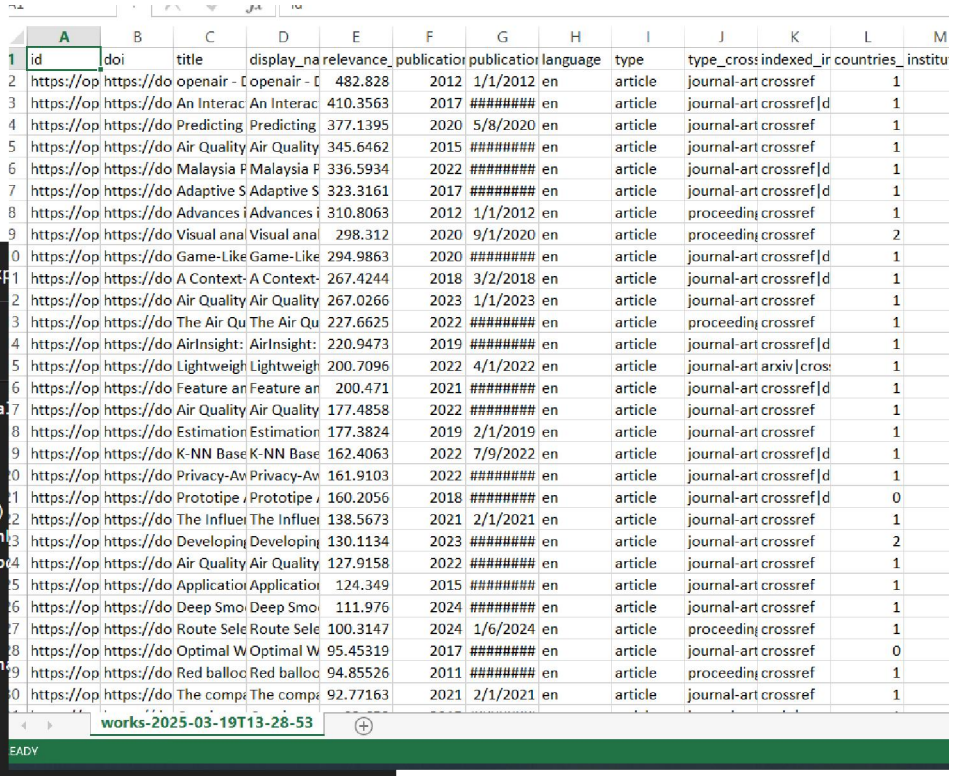




•**Selective Data Extraction via OpenAlex Web Interface:** Rather than downloading the full dataset, we leveraged the OpenAlex interface to perform targeted searches for works explicitly mentioning "air quality" within their titles and abstracts. We also conducted an additional targeted filtering for open-access works in English specifically focused on sustainability-related topics within the computer science field, which also referenced "air quality".

•**Filtered Results and PDF URL Extraction:** The UI searches on the OpenAlex website provided us with a CSV file containing extensive metadata about each work, including the title, author information, publication details, and most importantly, direct links to the available PDFs.

•We developed scripts to efficiently retrieve and store only the PDF URLs, substantially reducing the dataset size and making data ingestion manageable within our budget and technical constraints.

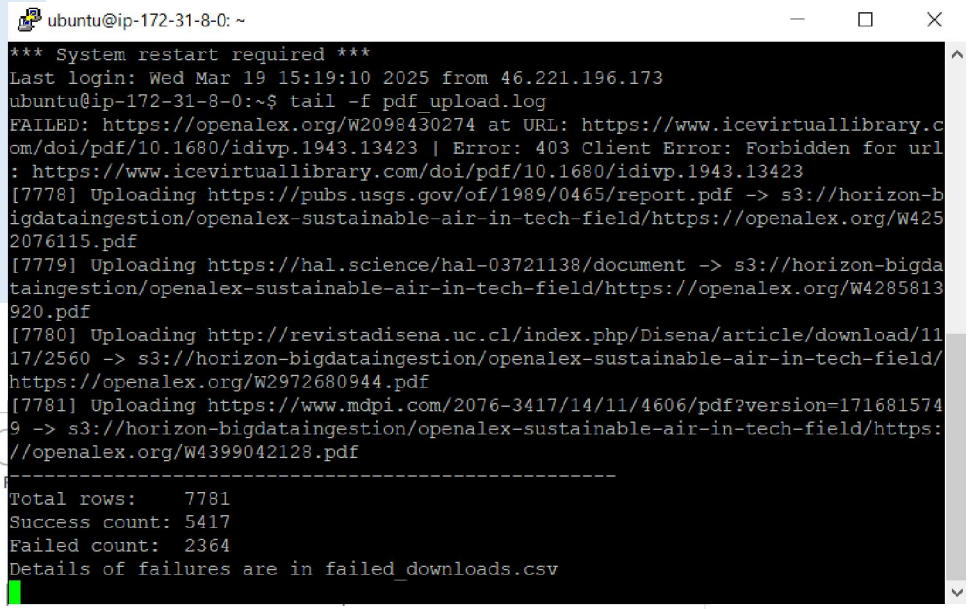


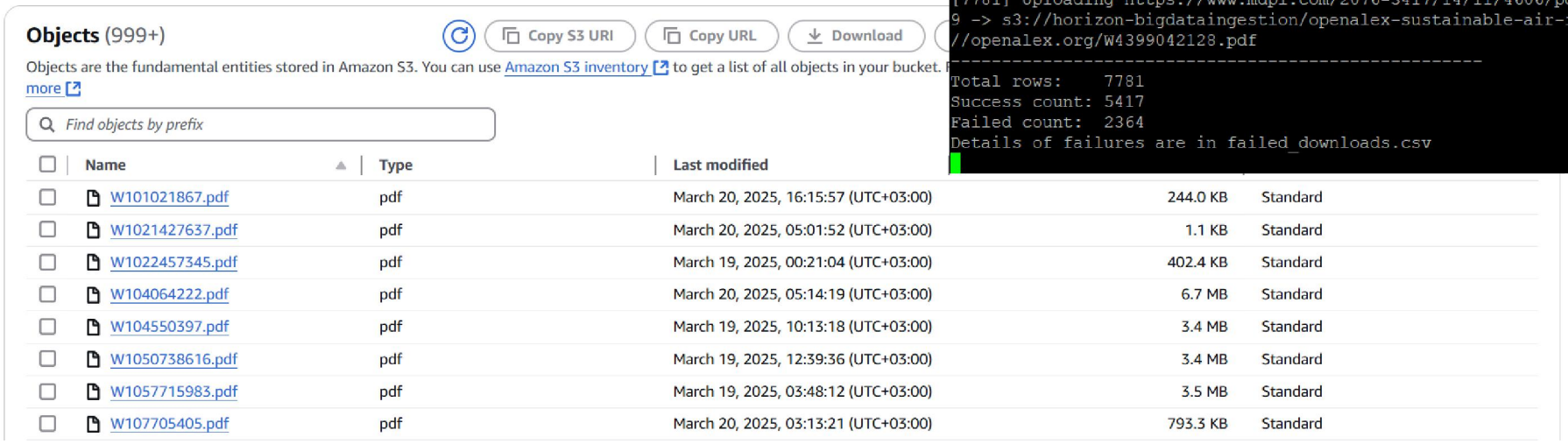
	A	B	C	D	E	F	G	H	I	J	K	L	M
1	id	doi	title	display_na	relevance_	publicatior	publicatior	language	type	type_cross	indexed_ir	countries_	institu
2	https://op	https://do	openair - [	openair - [	482.828	2012	1/1/2012	en	article	journal-art	crossref	1	
3	https://op	https://do	An Interac	An Interac	410.3563	2017	########	en	article	journal-art	crossref|d	1	
4	https://op	https://do	Predicting	Predicting	377.1395	2020	5/8/2020	en	article	journal-art	crossref	1	
5	https://op	https://do	Air Quality	Air Quality	345.6462	2015	########	en	article	journal-art	crossref	1	
6	https://op	https://do	Malaysia F	Malaysia F	336.5934	2022	########	en	article	journal-art	crossref|d	1	
7	https://op	https://do	Adaptive S	Adaptive S	323.3161	2017	########	en	article	journal-art	crossref|d	1	
8	https://op	https://do	Advances i	Advances i	310.8063	2012	1/1/2012	en	article	proceedin	crossref	1	
9	https://op	https://do	Visual anal	Visual anal	298.312	2020	9/1/2020	en	article	proceedin	crossref	2	
10	https://op	https://do	Game-Like	Game-Like	294.9863	2020	########	en	article	journal-art	crossref|d	1	
11	https://op	https://do	A Context-	A Context-	267.4244	2018	3/2/2018	en	article	journal-art	crossref|d	1	
12	https://op	https://do	Air Quality	Air Quality	267.0266	2023	1/1/2023	en	article	journal-art	crossref	1	
13	https://op	https://do	The Air Qu	The Air Qu	227.6625	2022	########	en	article	proceedin	crossref	1	
14	https://op	https://do	AirInsight:	AirInsight:	220.9473	2019	########	en	article	journal-art	crossref|d	1	
15	https://op	https://do	Lightweigh	Lightweigh	200.7096	2022	4/1/2022	en	article	journal-art	arxiv|cros	1	
16	https://op	https://do	Feature an	Feature an	200.471	2021	########	en	article	journal-art	crossref|d	1	
17	https://op	https://do	Air Quality	Air Quality	177.4858	2022	########	en	article	journal-art	crossref	1	
18	https://op	https://do	Estimation	Estimation	177.3824	2019	2/1/2019	en	article	journal-art	crossref	1	
19	https://op	https://do	K-NN Base	K-NN Base	162.4063	2022	7/9/2022	en	article	journal-art	crossref|d	1	
20	https://op	https://do	Privacy-Aw	Privacy-Aw	161.9103	2022	########	en	article	journal-art	crossref|d	1	
21	https://op	https://do	Prototipe /	Prototipe /	160.2056	2018	########	en	article	journal-art	crossref|d	0	
22	https://op	https://do	The Influer	The Influer	138.5673	2021	2/1/2021	en	article	journal-art	crossref	1	
23	https://op	https://do	Developin	Developin	130.1134	2023	########	en	article	journal-art	crossref	2	
24	https://op	https://do	Air Quality	Air Quality	127.9158	2022	########	en	article	journal-art	crossref	1	
25	https://op	https://do	Applicatior	Applicatior	124.349	2015	########	en	article	journal-art	crossref	1	
26	https://op	https://do	Deep Smo	Deep Smo	111.976	2024	########	en	article	journal-art	crossref	1	
27	https://op	https://do	Route Sele	Route Sele	100.3147	2024	1/6/2024	en	article	proceedin	crossref	1	
28	https://op	https://do	Optimal W	Optimal W	95.45319	2017	########	en	article	journal-art	crossref	0	
29	https://op	https://do	Red balloc	Red balloc	94.85526	2011	########	en	article	proceedin	crossref	1	
30	https://op	https://do	The compa	The compa	92.77163	2021	2/1/2021	en	article	journal-art	crossref	1	

works-2025-03-19T13-28-53

EADY

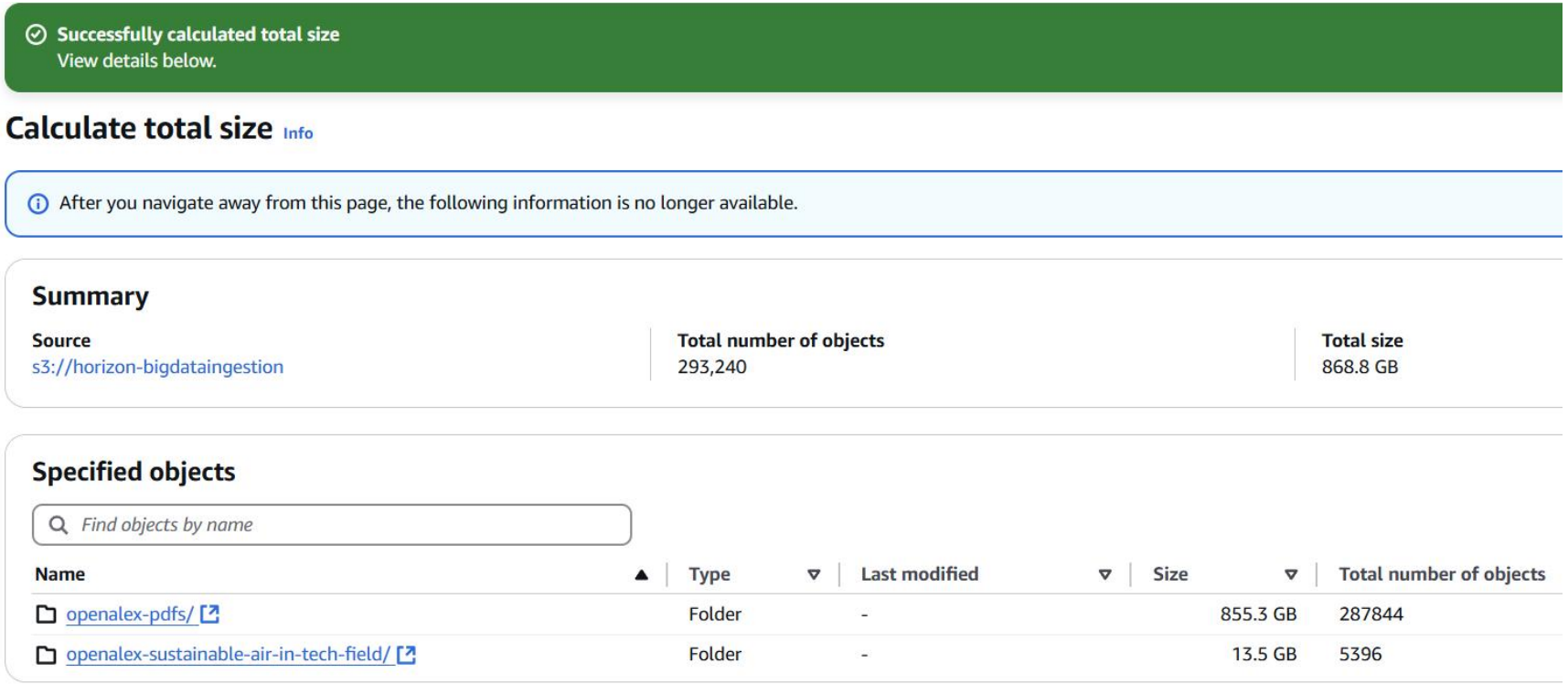
•**Optimized EC2-to-S3 Pipeline:** Filtered data (PDF URLs) were uploaded first to our own S3 bucket (horizon-bigdataingestion) and then read by our EC2 instances. Subsequently, we utilized small EC2 instances to download PDFs directly from these URLs and upload them into S3, handling errors gracefully by maintaining a list of failed downloads for debugging and potential retries.





Final Data Ingested and Stored for OpenAlex

Content:
- Total data size: ~868.8 GB
- Total number of PDFs: ~287,000+



Successfully calculated total size
View details below.

Calculate total size Info

After you navigate away from this page, the following information is no longer available.

Summary

Source	Total number of objects	Total size
s3://horizon-bigdataingestion	293,240	868.8 GB

Specified objects

Find objects by name

Name	Type	Last modified	Size	Total number of objects
openalex-pdfs/	Folder	-	855.3 GB	287844
openalex-sustainable-air-in-tech-field/	Folder	-	13.5 GB	5396

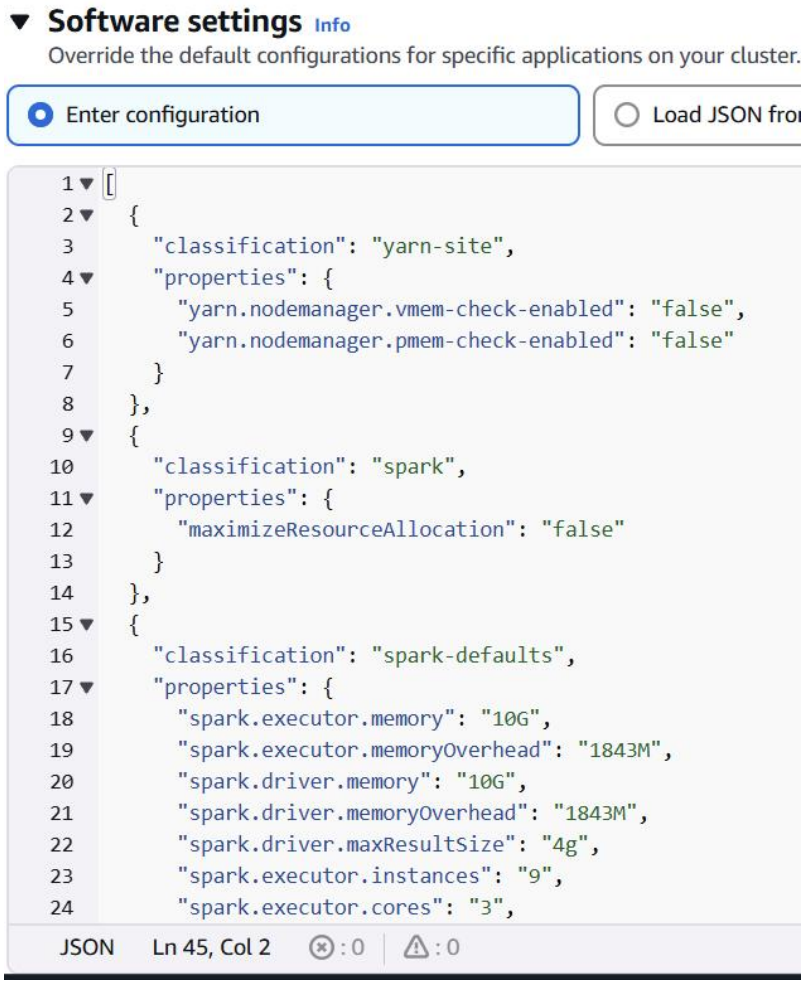
Processing OpenAlex PDFs

Overview

In this stage of the project, our objective was to transform a massive collection of OpenAlex PDF documents—originally stored in an S3 bucket—into a more analytics-friendly Parquet format. This would enable scalable and efficient querying for downstream analysis. The task required us to leverage Apache Spark on an AWS EMR cluster to parallelize processing. However, the pipeline posed unique challenges due to file formats, system constraints, and S3 structural anomalies.

Spark configuration optimizer

The very first step for a optimized spark job (as learned in the last lecture) was to add spark optimization configurations before launching the cluster. We used a 10 core cluster of m5.xlarge machines on a 5.36 version EMR and added a JSON file to the software settings in the cluster set-up.

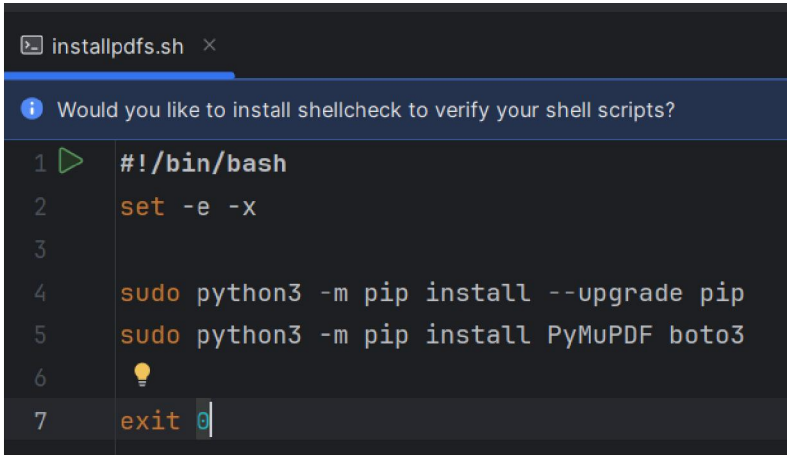


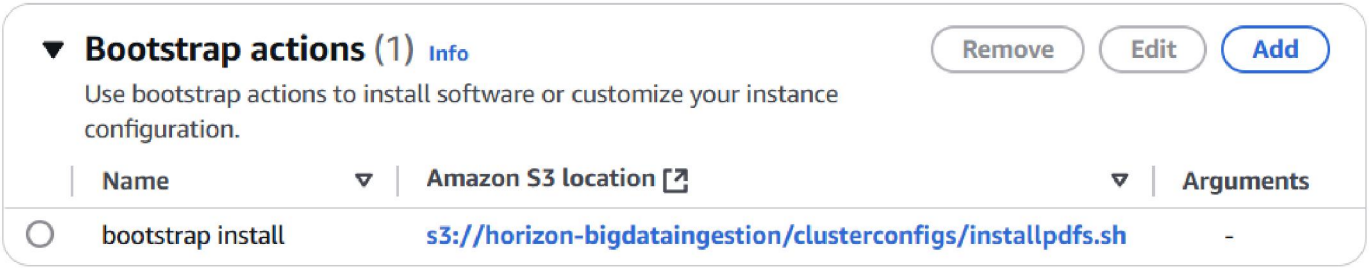
Challenges and Solutions

1. Spark's Inability to Natively Read PDFs

Apache Spark is not designed to parse binary formats like PDFs directly. To address this limitation, we adopted a custom parsing solution using the PyMuPDF library to extract text content from PDFs. This library was installed across all EMR nodes using *Bootstrap Actions*, ensuring consistency across the cluster.

Added the bootstrap .sh file to our s3 bucket and then added it to the cluster set-up.





2. Unconventional S3 Prefixes

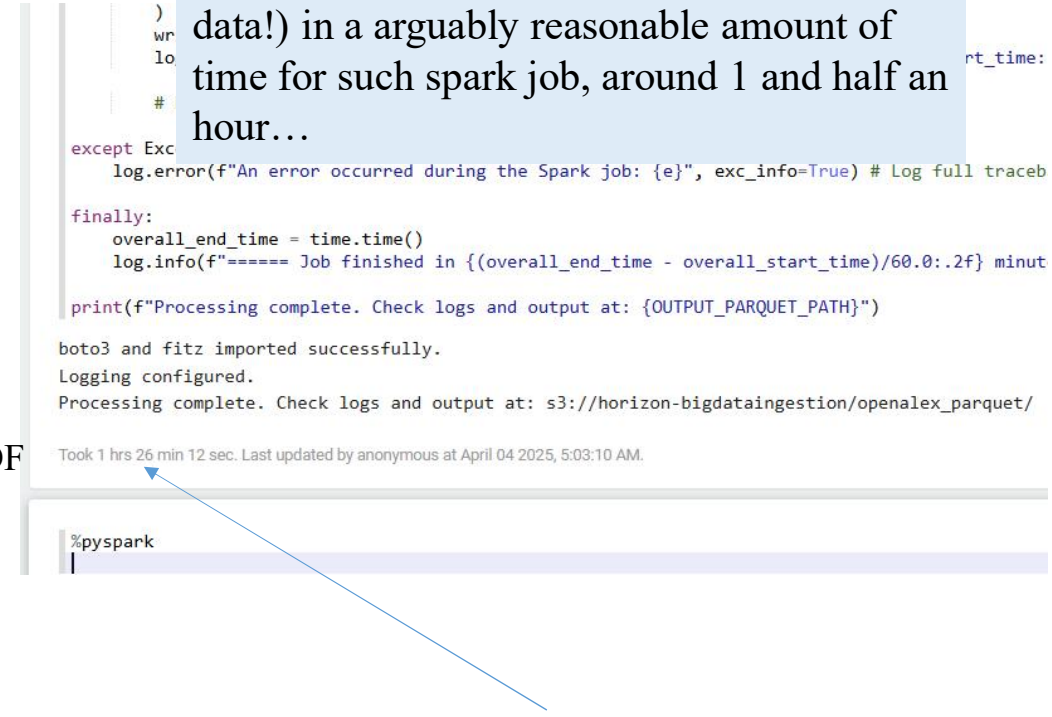
Due to ingestion artifacts, folder names included malformed characters like https: and /, which Spark interpreted as invalid URIs.
•**Issue**: Spark's file readers failed with URISyntaxExceptions.
•**Workaround**: We bypassed Spark's default S3 connector for file access. Instead, we used **boto3** (AWS SDK for Python) to directly fetch PDF files using their full S3 object keys within Spark tasks (mapPartitions).

3. Driver Memory Concerns

With over 200,000 PDF files totaling 855GB, scaling the solution introduced memory bottlenecks, particularly on the driver node.
•**Problem**: Listing and managing all file keys on the driver would exhaust memory.
•**Refined Approach**:

1. We first ran a lightweight Python script to list all valid S3 PDF keys and saved the list to a text file.
2. Then, the main Spark job loaded this list in parallel using textFile(), distributing the workload and fetching each PDF independently using boto3 inside the executors.

And it was with this optimized approach that we finally managed to parquet ~855gb of pdf data into a total of 10gb of parquet file, each file carefully enginnered to be around 128mg (or slightly more, which apparently is the sweet spot for spark and AWS when it comes for handling parquet data!) in a arguably reasonable amount of time for such spark job, around 1 and half an hour…



Key Spark Techniques & Commands Explained

1. RDD Creation and Parallelization

```
keys_rdd = spark.sparkContext.parallelize(pdf_s3_keys_list)
```

• **Why**: The list of S3 paths was created by the driver using boto3, then parallelized into an RDD to distribute processing across executors.
• **Impact**: Each partition handles a subset of keys, enabling efficient distributed text extraction from the PDFs.
What it does:This command takes a regular Python list—in this case, a list of full S3 paths to PDF files—and distributes it across the Spark cluster by turning it into a Resilient Distributed Dataset (RDD).💡 Why we used it:The list of S3 keys was generated using boto3 on the driver node, which isn't designed to handle the heavy lifting of parallel processing. By converting the list into an RDD, we handed off this workload to the Spark executors, allowing thousands of PDFs to be processed in parallel.⚙ How it helped:Each partition of the RDD receives a chunk of the file list, which the executors then process independently. This ensures distributed workload balance, leading to faster processing and better utilization of the EMR cluster.

2. mapPartitions for Efficient Processing

```
extracted_text_rdd = keys_rdd.mapPartitions(extract_text_for_keys)
```

• **Why**: mapPartitions allows each worker node to reuse resources like S3 clients and PyMuPDF parsers per partition, reducing overhead.
• **Impact**: Speeds up execution, reduces connection overhead, and supports scalable extraction of PDF text content.
What it does:The mapPartitions transformation applies the extract_text_for_keys function to entire partitions of data instead of one element at a time. Each partition is passed as an iterator.💡 Why we used it:When processing large-scale data like PDFs from S3, opening a new connection or re-importing libraries for each file (which happens with map()) would be extremely inefficient. Instead, mapPartitions allows us to:Create a single boto3 S3 client per executor task, reused for all files in that partition.Initialize and reuse PyMuPDF (fitz) instances efficiently.Avoid excessive overhead from repeatedly setting up resources.⚙ How it helped:This method dramatically improved performance and stability, especially in a job that interacts with external storage (S3) and performs CPU-heavy PDF parsing. It also lowered the risk of hitting rate limits or timeouts from repeated S3 calls.

3. RDD to DataFrame Conversion

```
schema = ["file_path", "content"]
extracted_df = extracted_text_rdd.toDF(schema)
```

- **Why:** DataFrames support optimized query planning, columnar storage, and structured transformation pipelines.
- **Impact:** Converts raw tuples into a schema-based structure ideal for Parquet writing.

What it does:This converts a raw RDD of tuples (S3 file path, extracted text) into a Spark DataFrame with named columns, which brings in the power of Spark SQL and optimized data transformations.💡 Why we used it:DataFrames are the de facto standard for structured data processing in Spark. They:Enable Catalyst optimization, which improves query and transformation performance.Support schema validation, making data easier to debug and maintain.Integrate seamlessly with Parquet writing, allowing Spark to handle columnar compression, statistics, and partition pruning.⚙ How it helped:Using a DataFrame allowed us to leverage Spark's columnar engine for high-speed Parquet generation and future queries. It also gave us readable, schema-based output that simplifies downstream processing.

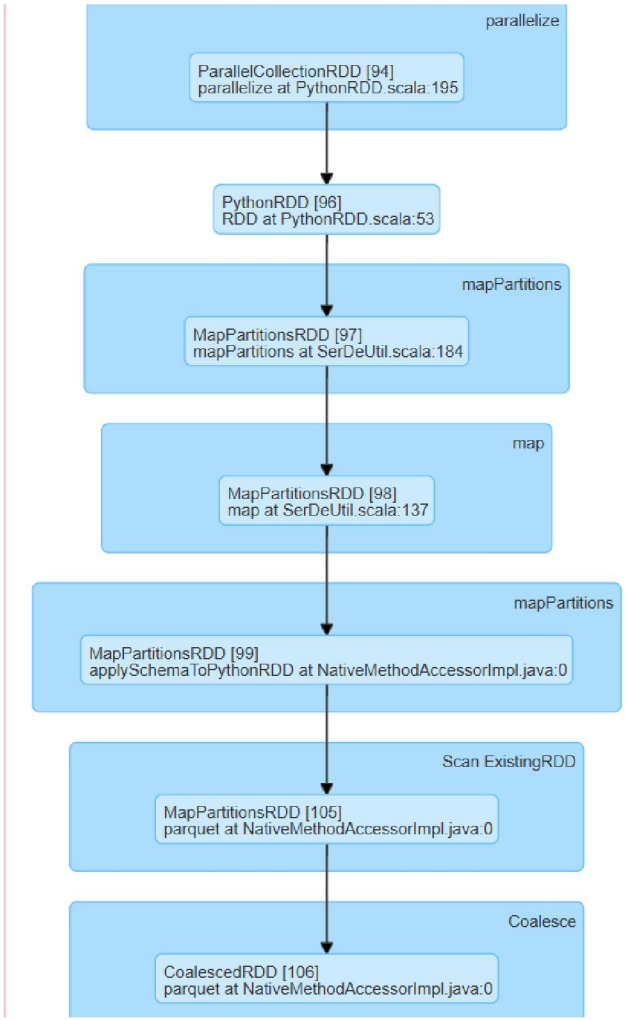
4. Coalescing Partitions for Output

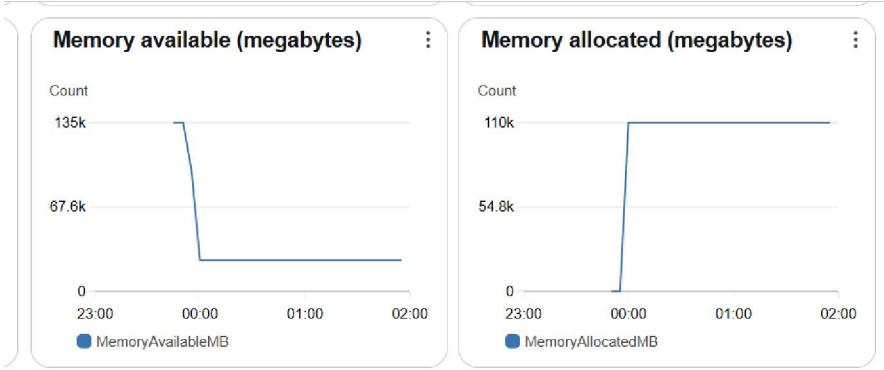
```
final_df = extracted_df.coalesce(OUTPUT_PARTITIONS)
```

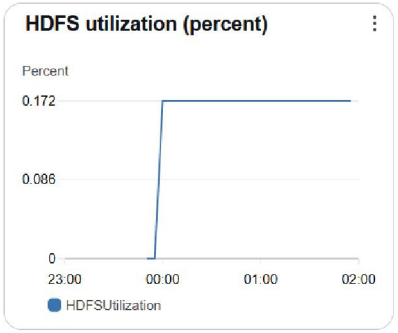
- **Why:** Coalescing reduces the number of output partitions without shuffling data—important for avoiding too many small Parquet files.Strategy: Targeted ~128–256MB per file. With ~300GB expected output, 2000 partitions was a safe starting estimate.
- **Impact:** Reduced strain on HDFS/S3, optimized performance during writes, and improved query efficiency downstream.

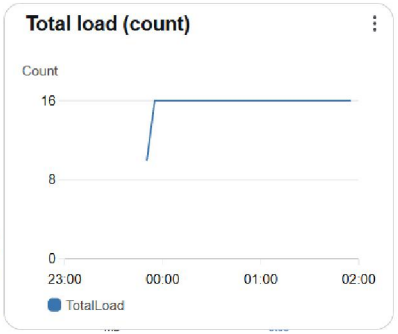
What it does:This transformation reduces the number of partitions in a DataFrame without triggering a full data shuffle. Unlike repartition(), coalesce() only merges adjacent partitions and avoids unnecessary data movement.💡 Why we used it:By default, Spark can produce an excessive number of output files—sometimes hundreds of thousands—each corresponding to a partition. This can:Slow down the write process.Overload S3 or HDFS with tiny files.Degrade query performance later due to small file problems.We used coalesce() to control and reduce the number of output files to around 2000, each ideally between 128MB and 256MB, which is considered optimal for most Parquet-based big data systems.⚙ How it helped:Fewer output files meant more manageable storage.Faster write times and better use of executor memory.Improved downstream analytics performance on tools like AWS Athena or Spark SQL.

DAG visualization and Insights of Cluster Utilization Monitorings





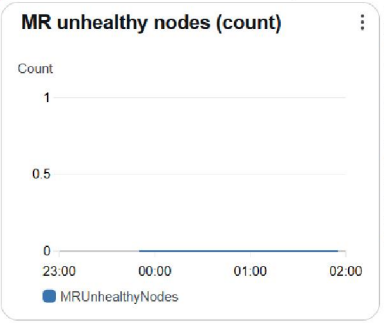




Executors

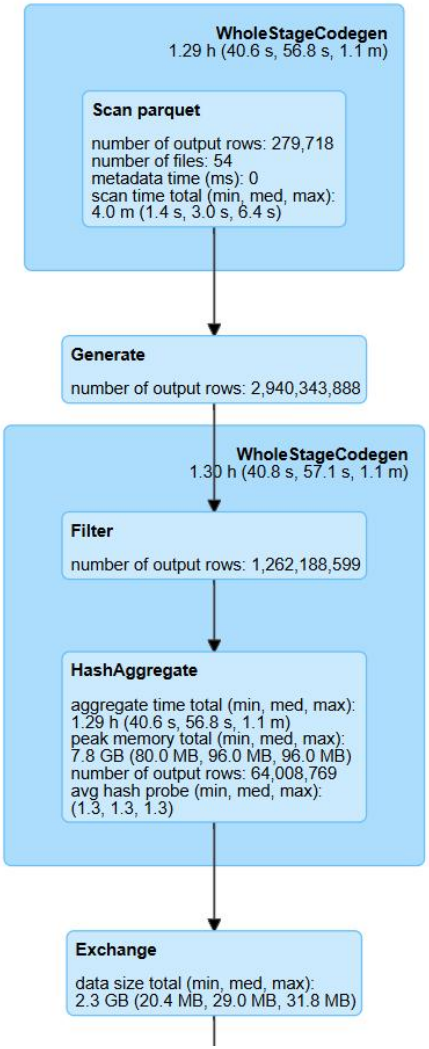
Show 100 entries

Executor ID	Address	Status	RDD Blocks	Storage Memory	On Heap Storage Memory	Off Heap Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	T T				
driver	ip-10-0-124-45.eu-central-1.compute.internal:46229	Active	0	0.0 B / 7.4 GB	0.0 B / 7.4 GB	0.0 B / 0.0 B	0.0 B	0	0	0	0	0				
1	ip-10-0-158-188.eu-central-1.compute.internal:41023	Active	0	0.0 B / 8.3 GB	0.0 B / 8.3 GB	0.0 B / 0.0 B	0.0 B	3	1	0	72	7				
2	ip-10-0-79-124.eu-central-1.compute.internal:33299	Active	0	0.0 B / 8.3 GB	0.0 B / 8.3 GB	0.0 B / 0.0 B	0.0 B	3	2	0	73	75	2.1 h (2 s)	95.2 MB	9.9 MB	14 MB	stdo stde
3	ip-10-0-1-177.eu-central-2333	Active	0	0.0 B / 8.3 GB	0.0 B / 8.3 GB	0.0 B / 0.0 B	0.0 B	3	1	0	72	73	2.2 h (2 s)	93.7 MB	13.9 MB	12.6 MB	stdo stde
	entral-5125	Active	0	0.0 B / 8.3 GB	0.0 B / 8.3 GB	0.0 B / 0.0 B	0.0 B	3	2	0	71	73	2.2 h (2 s)	92.7 MB	13.8 MB	12.8 MB	stdo stde
	ntral-6337	Active	0	0.0 B / 8.3 GB	0.0 B / 8.3 GB	0.0 B / 0.0 B	0.0 B	3	2	0	75	77	1.9 h (2 s)	100.9 MB	11.5 MB	12.9 MB	stdo stde
	central-6141	Active	0	0.0 B / 8.3 GB	0.0 B / 8.3 GB	0.0 B / 0.0 B	0.0 B	3	1	0	74	75	2.3 h (2 s)	93.1 MB	13.9 MB	12.4 MB	stdo stde
	central-4619	Active	0	0.0 B / 8.3 GB	0.0 B / 8.3 GB	0.0 B / 0.0 B	0.0 B	3	3	0	66	69	1.7 h (2 s)	92.3 MB	13.9 MB	12.8 MB	stdo stde
	central-2713	Active	0	0.0 B / 8.3 GB	0.0 B / 8.3 GB	0.0 B / 0.0 B	0.0 B	3	2	0	72	74	1.9 h (1 s)	92.9 MB	12.2 MB	12.9 MB	stdo stde
	entral-9647	Active	0	0.0 B / 8.3 GB	0.0 B / 8.3 GB	0.0 B / 0.0 B	0.0 B	3	1	0	71	72	2.3 h (2 s)	98.5 MB	13.8 MB	12.7 MB	stdo stde



OPENALEX Analysis Workflow Overview

A multi-step process was implemented using PySpark to handle the large dataset efficiently and get some insights from the openalex parquet data :



Key Step 1 & 2: Data Loading & Preprocessing
Details:

Read all Parquet files from the specified S3 input path.
Convert all text content to lowercase for case-insensitive counting.
Remove punctuation, symbols, and non-alphanumeric characters to isolate words.
Remove specific PDF extraction artifacts (e.g., ---, page break).

Key Spark Commands:

spark.read.parquet(PATH): Efficiently loads structured Parquet data into a Spark DataFrame.
F.lower(column): Converts a text column to lowercase.
F.regexp_replace(column, pattern, replacement): Uses regular expressions to find and replace patterns (e.g., removing [^a-z0-9\s] keeps only letters, numbers, whitespace).

Step 3: Tokenization
Details:

Split the preprocessed text strings into arrays of words based on whitespace.
Transform the DataFrame so each row contains only a single word. This is crucial for counting individual word frequencies.

Key Spark Commands:

F.split(column, pattern): Splits a string column into an array based on a delimiter (e.g., \s+ for one or more spaces).
F.explode(array_column): Creates a new row for each element in an array column (turning the word array into individual word rows).

Key Step 4: Filtering (Stop Word Removal)

- **Details:**
  - Filter out any empty strings generated during preprocessing.
  - Filter out strings containing only numbers.
  - Define a comprehensive list (STOP_WORDS) containing:
    - Standard English stop words (e.g., "the", "a", "is").
    - General academic terms (e.g., "figure", "table", "references").
    - Common units, locations, names.
    - General science terms not specific to the target technical domain (e.g., "water", "air").
  - Filter the DataFrame, keeping only words *not* present in the stop word list.
- **Key Spark Commands:**
  - df.filter(condition): Selects rows that meet specific criteria (e.g., word != "", not purely numeric ~word.rlike("^[0-9]+$")).
  - spark.sparkContext.broadcast(value): Efficiently sends a read-only copy of the large stop word list to all worker nodes. Avoids redundant network traffic.
  - ~F.col(column).isin(broadcast_variable.value): Filters rows where the column value is *not* present in the broadcasted stop word set.

- **Key Step 5 & 6: Word Counting & Ranking/Selection**
- **Details:**
  - Group the filtered words together.
  - Count the number of occurrences for each unique word.
  - Sort the results in descending order based on the count.
  - Select only the top 100 rows from the sorted list.
- **Key Spark Commands:**
  - df.groupBy("word").count(): Performs the core aggregation – groups rows by the 'word' column and counts the size of each group.
  - df.orderBy(F.col("count").desc()): Sorts the DataFrame based on the 'count' column in descending order.
  - df.limit(100): Reduces the DataFrame to only the first 100 rows (the top 100 most frequent words after sorting).

Key Step 7: Output Generation

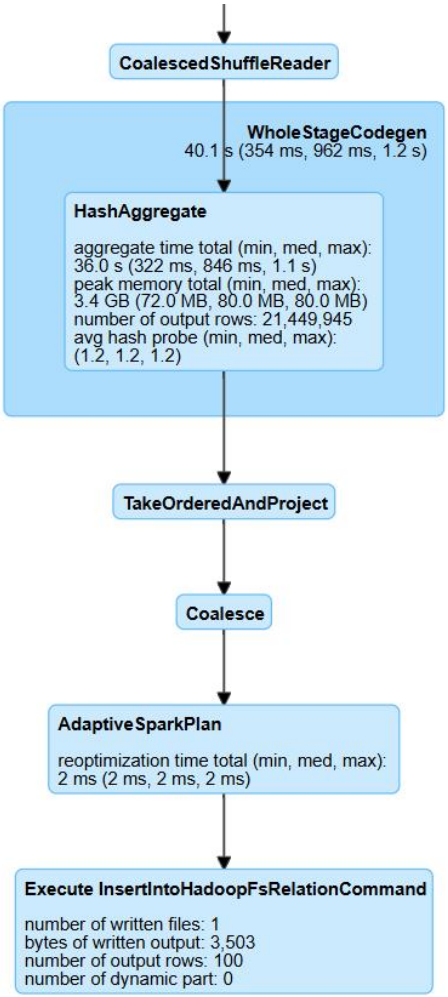
•Details:

- Combine the results into a single partition to ensure a single output file.
- Write the resulting DataFrame to the specified S3 location in JSON format.
- Configure the write operation to overwrite any existing data at the destination.

•Key Spark Commands:

- df.coalesce(1): Reduces the number of DataFrame partitions to one. Essential for generating a single output file (but can impact performance for very large results).
- df.write.mode("overwrite").json(PATH): Writes the DataFrame to the specified PATH.
  - .write: Accesses the DataFrameWriter API.
  - .mode("overwrite"): Specifies that existing data at the path should be replaced.
  - .json(): Sets the output format to JSON (one JSON object per row/word).

- **Key Step 7: Output Generation**
- **Details:**
  - Combine the results into a single partition to ensure a single output file.
  - Write the resulting DataFrame to the specified S3 location in JSON format.
  - Configure the write operation to overwrite any existing data at the destination.
- **Key Spark Commands:**
  - df.coalesce(1): Reduces the number of DataFrame partitions to one. Essential for generating a single output file (but can impact performance for very large results).
  - df.write.mode("overwrite").json(PATH): Writes the DataFrame to the specified PATH.
    - .write: Accesses the DataFrameWriter API.
    - .mode("overwrite"): Specifies that existing data at the path should be replaced.
    - .json(): Sets the output format to JSON (one JSON object per row/word).



Final Output

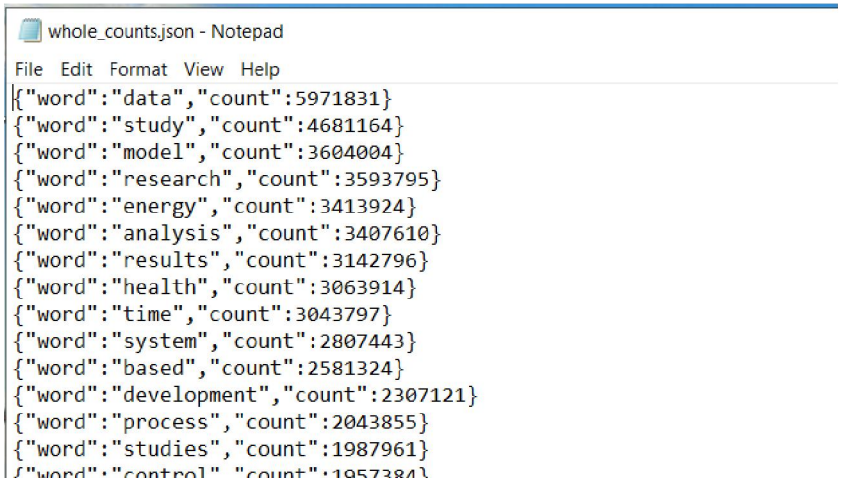
Output: A single JSON file located in s3://horizon-bigdataingestion/openalex_jsons/.

•**Content:** Contains the top 100 most frequent words (identified as key technical concepts) and their corresponding counts.

•**Format:** Each line is a JSON object: {"word": "concept_word", "count": 12345}

•**Purpose:** This structured JSON output is ideal for:

- •Easy ingestion into Elasticsearch.
- •Subsequent visualization in Kibana (e.g., creating word clouds, bar charts of top terms) to explore trends and themes within the OpenAlex dataset.


whole_counts.json - Notepad

File Edit Format View Help

{"word":"data","count":5971831}
{"word":"study","count":4681164}
{"word":"model","count":3604004}
{"word":"research","count":3593795}
{"word":"energy","count":3413924}
{"word":"analysis","count":3407610}
{"word":"results","count":3142796}
{"word":"health","count":3063914}
{"word":"time","count":3043797}
{"word":"system","count":2807443}
{"word":"based","count":2581324}
{"word":"development","count":2307121}
{"word":"process","count":2043855}
{"word":"studies","count":1987961}


```
    # --- Step 6: Save Top 100 Results to a Single JSON File ---
    print("\n--- Step 6: Saving top 100 word counts to a single JSON file ---")
    print(f"Target S3 folder: {JSON_OUTPUT_FOLDER}")

    # Use coalesce(1) to ensure the output is written to a single file
    # Use mode("overwrite") to replace the target folder if it exists
    top_100_words_df.coalesce(1).write.mode("overwrite").json(JSON_OUTPUT_FOLDER)

    print(f"Successfully saved top 100 word counts to: {JSON_OUTPUT_FOLDER}")

except Exception as e:
    # Catch and report any errors that occur during execution
    print(f"\n--- ERROR ---")
    print(f"An error occurred: {e}")
    import traceback
    traceback.print_exc()

# Indicate successful completion of the script
print("\n--- Script Finished ---")
```

```
|capacity       |679771 |
|reduce         |678454 |
|detection      |657248 |
|reduced        |649440 |
|age            |649249 |
|loss           |647997 |
|relationship   |632351 |
|selected       |630933 |
|ing            |630440 |
|correlation    |626526 |
+---------------+-------+


--- Step 6: Saving top 100 word counts to a single JSON file ---
Target S3 folder: s3://horizon-bigdataingestion/openalex_jsons/
Successfully saved top 100 word counts to: s3://horizon-bigdataingestion/openalex_jsons/

--- Script Finished ---
```

Took 6 min 57 sec. Last updated by anonymous at April 09 2025, 9:51:05 PM.

Year Count Job Documentation

•**Goal:** To quantify the occurrences of specific calendar years (e.g., 1995, 2020) mentioned within the OpenAlex PDF text corpus.

Key Step: Year Filtering

•**Details:**

- After basic cleaning and tokenization, apply filters:
  - Identify tokens consisting of exactly four digits (^[0-9]{4}$).
  - Convert these 4-digit tokens to integers.
  - Keep only those integer values falling between the predefined MIN_YEAR (1900) and MAX_YEAR (2050).

•**Key Spark Commands:**

- F.col("token").rlike("^[0-9]{4}$"): Filters tokens matching the 4-digit pattern using regular expressions.
- .cast(IntegerType()): Converts the string token to a numerical integer type.
- F.col("year_int").between(MIN_YEAR, MAX_YEAR): Selects rows where the integer value is within the specified range.

Key Step: Counting & Output

•**Details:**

- Group the filtered DataFrame by the validated 'year' string.
- Count the occurrences for each unique year.
- Order the results chronologically by year (ascending).
- Combine results into a single partition.
- Write the data to a *new, dedicated* S3 folder (.../openalex_json/) as a single JSON file, overwriting previous content in that specific folder if any.

•**Key Spark Commands:**

- df.groupBy("year").count(): Aggregates counts for each unique year.
- df.orderBy(F.col("year").asc()): Sorts the results chronologically.
- df.coalesce(1): Ensures output is written to a single file.
- df.write.mode("overwrite").json(PATH): Saves the result as JSON to the designated path.

Year Count Output

•**Output Location:** s3://horizon-bigdataingestion/openalex_year_counts_json/

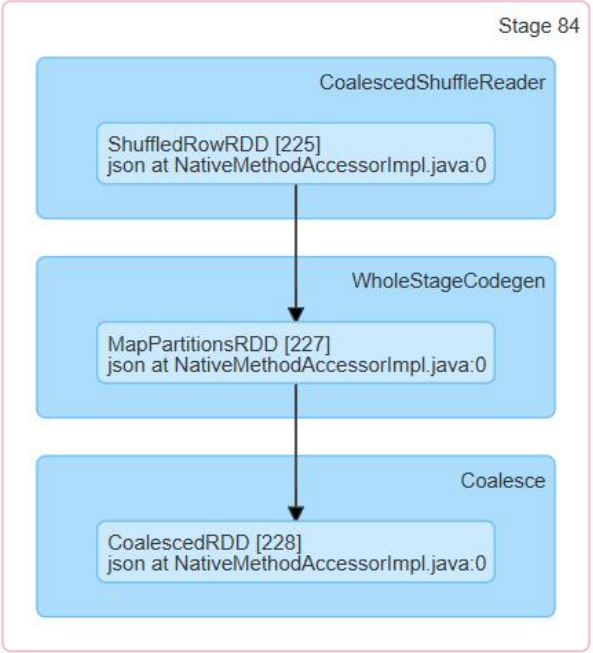
•**Format:** Single JSON file containing objects like:

year_counts.json - Notepad

File Edit Format View Help


```
{"year":"1900","count":23555}
{"year":"1901","count":6693}
{"year":"1902","count":5280}
{"year":"1903","count":5646}
{"year":"1904","count":5626}
{"year":"1905","count":6866}
{"year":"1906","count":5707}
{"year":"1907","count":5656}
```


•**Purpose:** Provides data ready for visualizing temporal trends (e.g., line chart of mentions per year) in Kibana or other tools.




```
    print("\n--- Step 6: Saving year counts to a single JSON file ---")
    print(f"Target S3 folder: {JSON_OUTPUT_FOLDER}")

    # Use coalesce(1) to ensure the output is written to a single file
    # Use mode("overwrite") to replace the target folder if it exists
    ordered_year_counts_df.coalesce(1).write.mode("overwrite").json(JSON_OUTPUT_FO

    print(f"Successfully saved year counts to: {JSON_OUTPUT_FOLDER}")

except Exception as e:
    # Catch and report any errors that occur during execution
    print(f"\n--- ERROR ---")
    print(f"An error occurred: {e}")
    import traceback
    traceback.print_exc()

# Indicate successful completion of the script
print("\n--- Script Finished ---")
```

```
|1942|9310 |
|1943|9590 |
|1944|9392 |
|1945|17699|
|1946|13198|
|1947|10229|
|1948|14464|
|1949|13432|
+----+-----+
only showing top 50 rows


--- Step 6: Saving year counts to a single JSON file ---
Target S3 folder: s3://horizon-bigdataingestion/openalex_jsons/
Successfully saved year counts to: s3://horizon-bigdataingestion/openalex_jsons/

--- Script Finished ---
```

Took 5 min 12 sec. Last updated by anonymous at April 09 2025, 10:10:56 PM.

Pollutant Count Job Documentation

•**Goal:** To count the frequency of mentions for a predefined list of specific environmental pollutants (e.g., 'co2', 'pm25', 'lead', 'benzene') within the OpenAlex text corpus.

Key Step: Text Preprocessing for Matching

•**Details:**

- Convert all text to lowercase.
- Remove all characters *except* lowercase letters (a-z), numbers (0-9), and whitespace. This is critical for ensuring terms like PM2.5 become pm25 and can be matched against a list containing pm25.

•**Key Spark Commands:**

- F.lower(column): Case normalization.
- F.regexp_replace(column, r'[^a-z0-9\s]', ''): Removes punctuation and symbols, standardizing tokens.

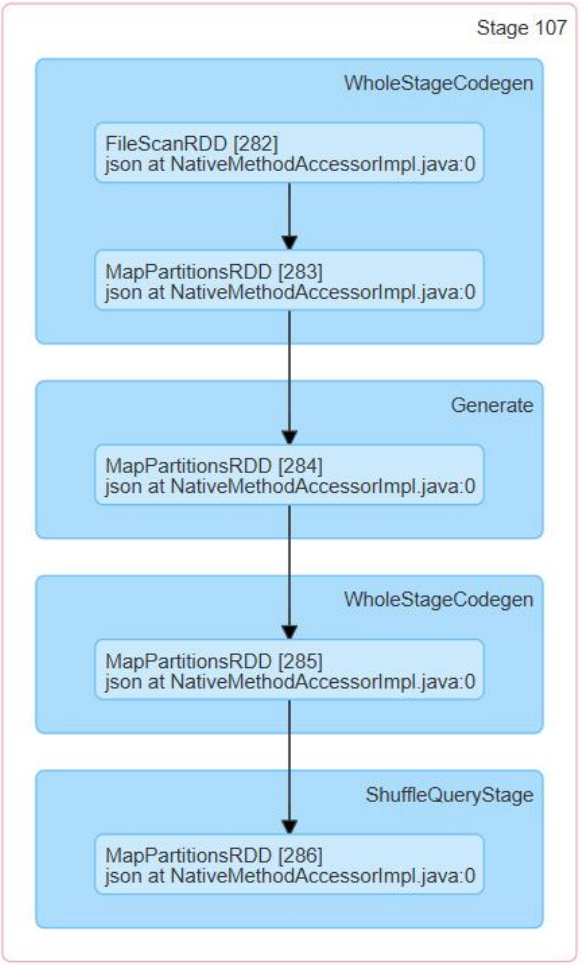
Key Step: Filtering for Pollutants

•**Details:**

- Define a Python list (POLLUTANT_LIST) containing all target pollutant terms (names, formulas) in lowercase.
- Convert this list to a set for efficient lookup.
- Broadcast the pollutant set to all Spark worker nodes.
- Filter the token DataFrame, keeping only rows where the token is found within (isin) the broadcasted set.

•**Key Spark Commands:**

- POLLUTANT_LIST = [...]: Defines the target terms.
- spark.sparkContext.broadcast(set(POLLUTANT_LIST)): Efficiently distributes the lookup set.
- df.filter(F.col("token").isin(broadcast_variable.value)): Selects rows matching any term in the set.



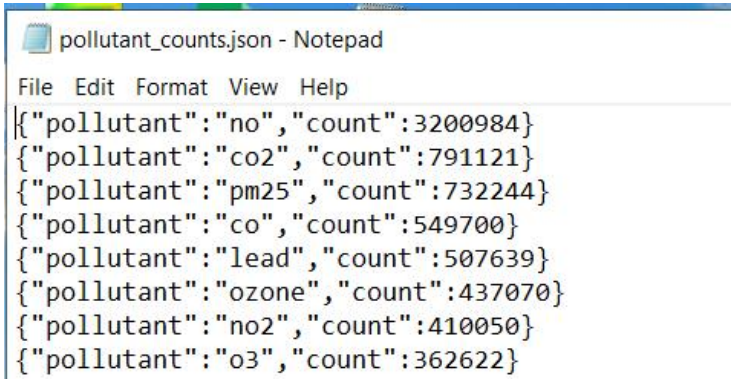
Slide 6: Key Step: Counting & Output

•Details:

- Group the filtered DataFrame by the matched 'pollutant' string.
- Count the occurrences for each unique pollutant found.
- Order the results by count (descending) to see the most frequently mentioned.
- Combine results into a single partition.
- Write the data to a *new, dedicated* S3 folder (.../openalex_pollutant_counts_json/) as a single JSON file, overwriting previous content in that specific folder if any.

•Key Spark Commands:

- df.groupBy("pollutant").count(): Aggregates counts for each matched pollutant.
- df.orderBy(F.col("count").desc()): Ranks pollutants by frequency.
- df.coalesce(1): Ensures single file output.
- df.write.mode("overwrite").json(PATH): Saves the result as JSON.



Pollutant Count Output & Caveats

Output Location: s3://horizon-bigdataingestion/openalex_pollutant_counts_json/

Format: Single JSON file containing objects like:

Caveat: Counts for very short, common terms that are *also* pollutant symbols (like 'no') might be inflated as the simple matching doesn't distinguish context (e.g., English word "no" vs. Nitric Oxide "NO"). Counts for less ambiguous terms are reliable.

Purpose: Provides data ready for analyzing the focus on specific pollutants (e.g., bar chart of top pollutant mentions) in Kibana.

2. OpenAQ

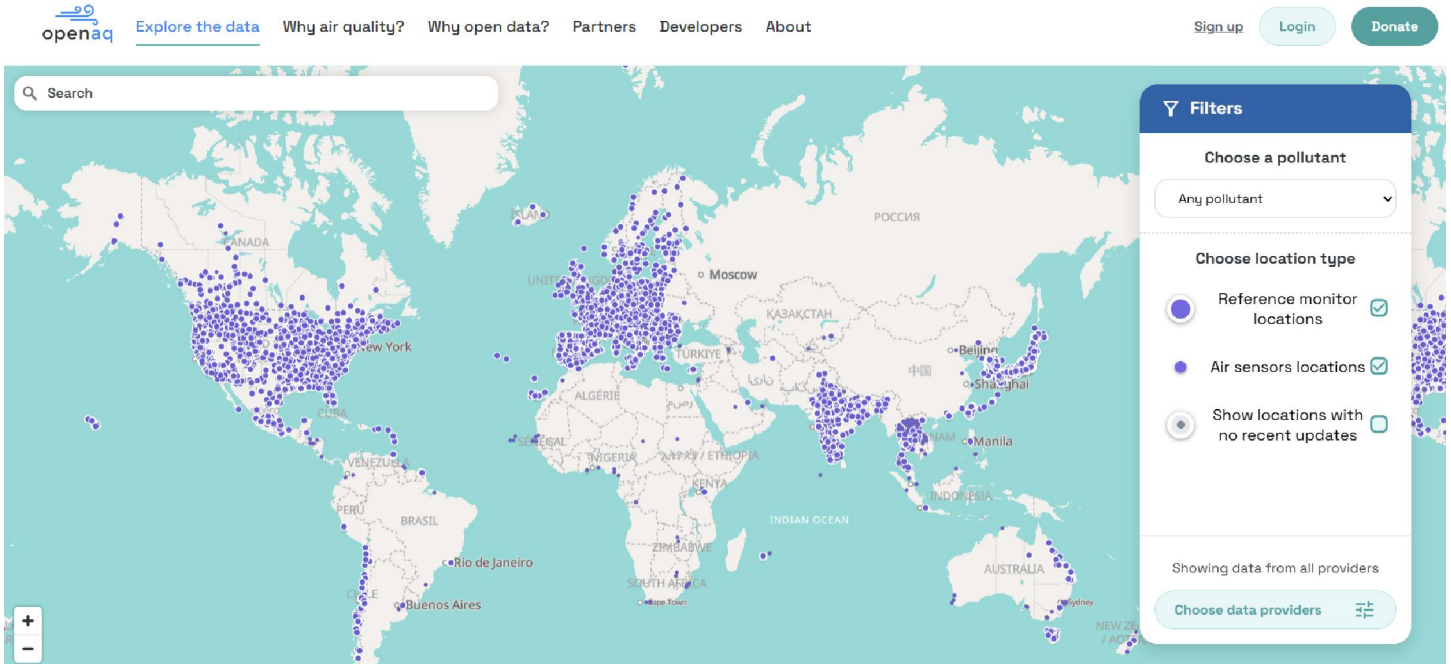
OpenAQ is an open-source environmental tech nonprofit platform that aggregates and shares historical air quality data from multiple sources around the world. It collects data from: Governmental air quality monitoring networks, Research institutions, Citizen science projects, IoT devices and low-cost sensors; into a single, open-source data platform. OpenAQ helps anyone—from scientists and policymakers to journalists and activists—analyze trends, communicate clearly, and advocate effectively for cleaner air. Their mission is to tackle "air inequality," the unequal access to clean air experienced around the world.

Vision, Mission and Values

OpenAQ actively advocates for increased air quality monitoring and greater data transparency, especially in marginalized and underserved communities. They uniquely publish the only global report assessing how countries publicly share air quality data.

OpenAQ's core values—**collaboration**, **equity**, **sustainability**, and **transparency**—guide their efforts to create a healthier world, driven not by profit but by the shared mission of cleaner air for all.

As mentioned in their "**OpenAQ's Origin Story**" on Medium, OpenAQ started from a simple yet powerful idea in late 2014, when atmospheric scientist Christa Hasenkopf struggled to find accessible and standardized air quality data for diplomatic staff at U.S. embassies. Together with her partner, technologist Joe Flasher, they recognized that open, real-time air quality data could wake the world up to air pollution and empower communities to fight air inequality. Over a single weekend, Joe built an open-source prototype using data from China, India, and Mongolia, which quickly grew into the largest global platform for open air quality data—driven entirely by volunteer contributions and an unwavering commitment to public good.

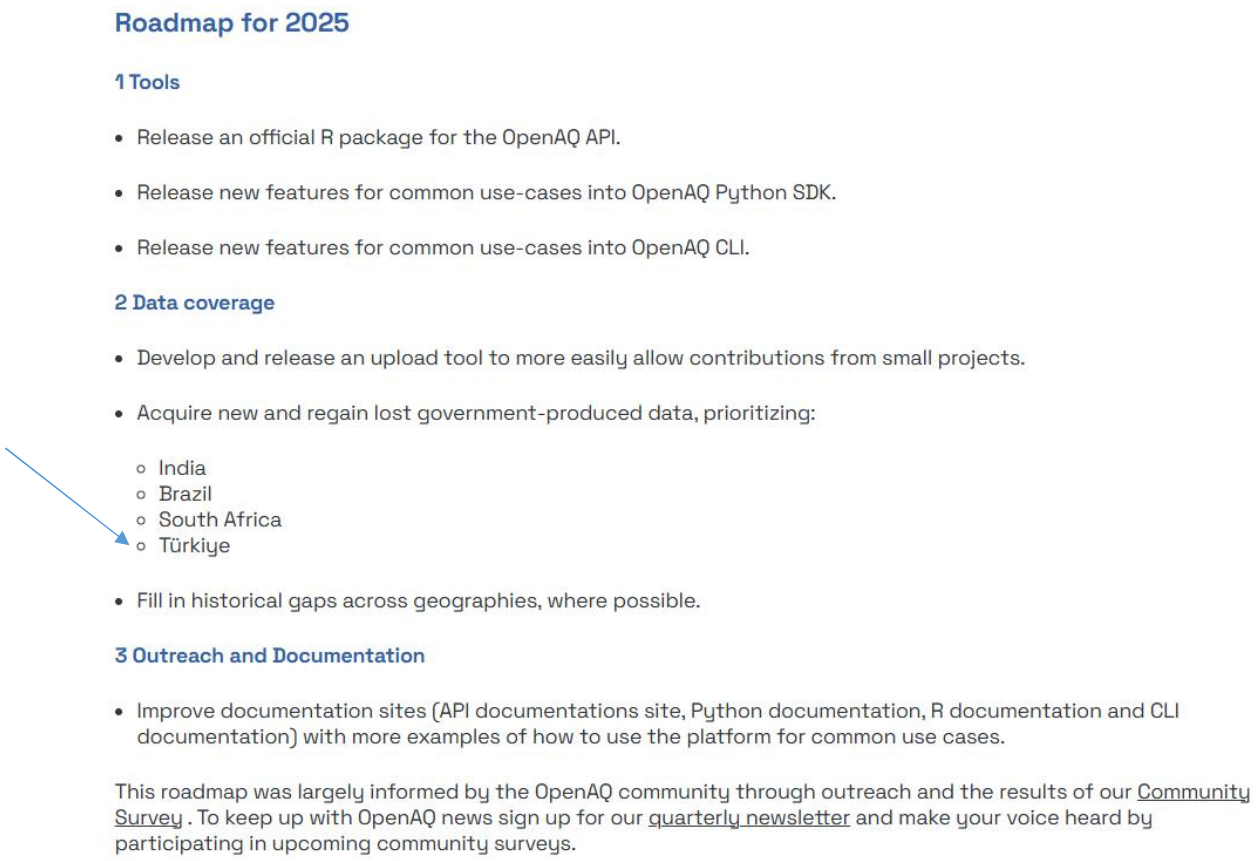


OpenAQ was founded on the principle that people have a right to know what they are breathing and should be able to freely access, use, reuse, and redistribute air quality data.

Inspired by OpenAQ: Our Shared Journey

Just like OpenAQ's founders, our journey began with an everyday experience—a sudden realization of the severity of air pollution around us. Discovering OpenAQ's story resonated deeply with us. They too started with frustration, determination, and the belief in the power of open, accessible data. Their commitment to transparency, equity, collaboration, and public good inspired our own path forward. Like them, we see open data not just as information, but as a powerful tool for innovation, accountability, and positive change.

We strongly value OpenAQ's principles—that clean air data should be openly accessible to empower communities, inform better decisions, and drive meaningful action. Motivated by their vision, our project aims to contribute back to the open-data ecosystem, supporting OpenAQ's ambitious sustainability goals for 2025…



Roadmap for 2025

1 Tools

- Release an official R package for the OpenAQ API.
- Release new features for common use-cases into OpenAQ Python SDK.
- Release new features for common use-cases into OpenAQ CLI.

2 Data coverage

- Develop and release an upload tool to more easily allow contributions from small projects.
- Acquire new and regain lost government-produced data, prioritizing:
  - India
  - Brazil
  - South Africa
  - Türkiye
- Fill in historical gaps across geographies, where possible.

3 Outreach and Documentation

- Improve documentation sites (API documentations site, Python documentation, R documentation and CLI documentation) with more examples of how to use the platform for common use cases.

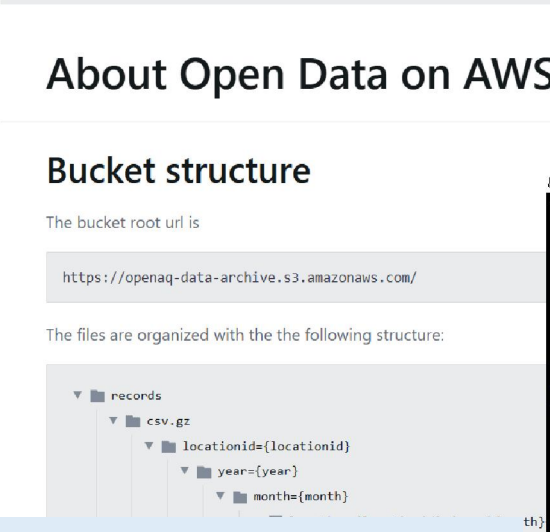
This roadmap was largely informed by the OpenAQ community through outreach and the results of our Community Survey . To keep up with OpenAQ news sign up for our quarterly newsletter and make your voice heard by participating in upcoming community surveys.

When exploring the OpenAQ Roadmap for 2025, we noticed an important gap: Turkey's air quality data coverage was identified as a priority area needing improvement.
As a part of our own journey, we successfully collected historical and real-time air quality data from Turkey by scraping the government's official air quality monitoring platform, **Ulusal Hava Kalite İzleme Ağı**. Realizing we had precisely what OpenAQ needed, we decided that giving back to the community would be one of our project's core future goals.

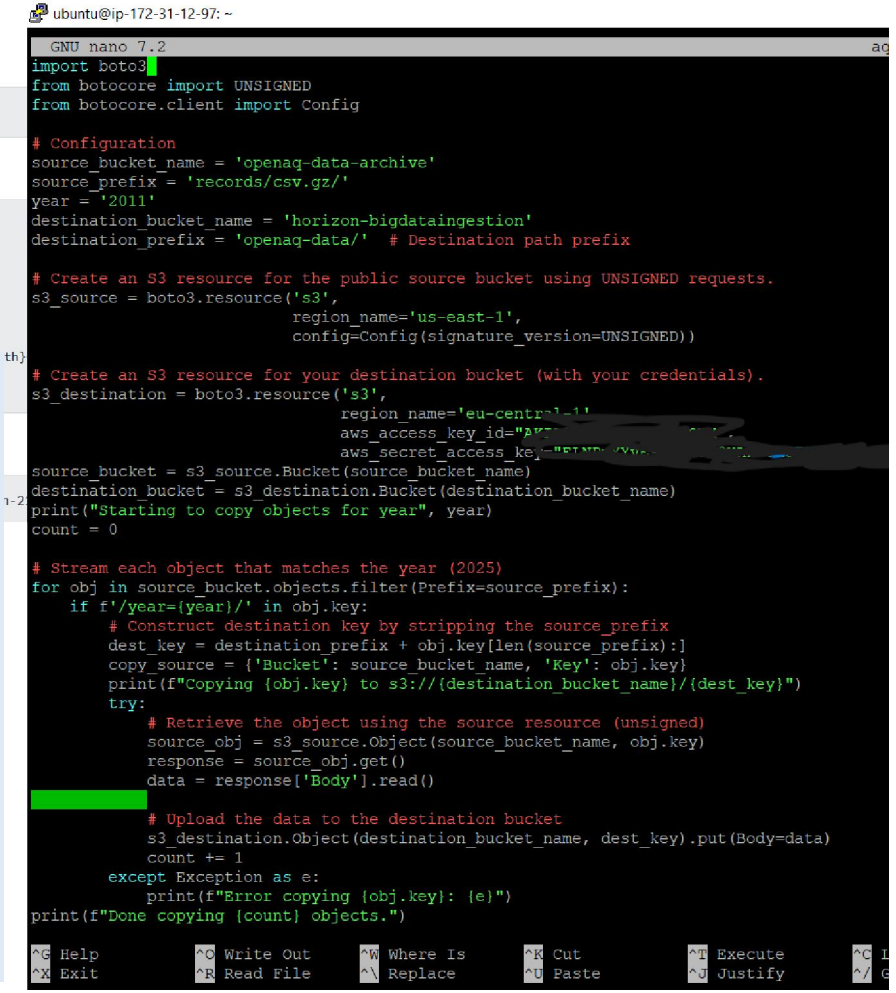
Through this contribution, we join OpenAQ in the larger global effort for open, accessible environmental data and sustainability.

Ingesting OpenAQ Data

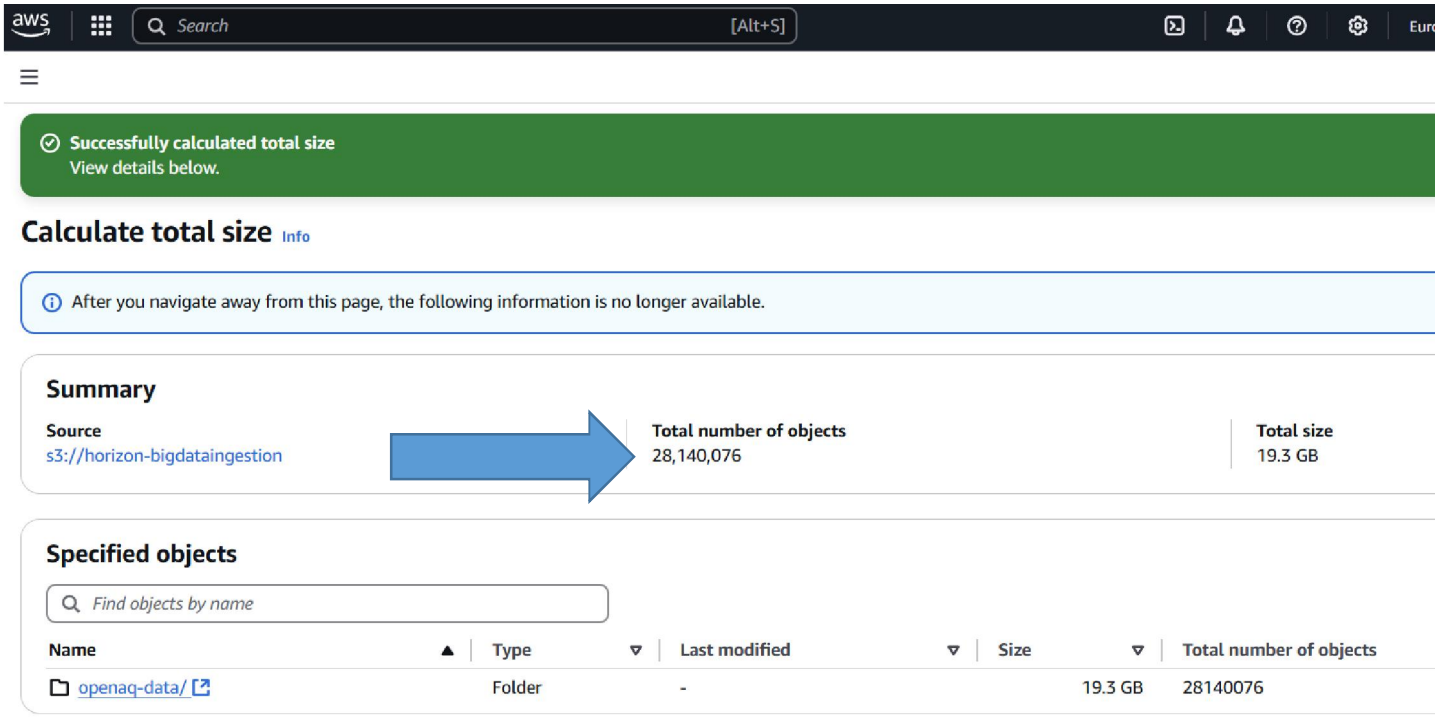
OpenAQ provides multiple ways to access their extensive air quality database:
•An interactive user interface (OpenAQ Explorer)
•A robust, publicly available REST API
•An open-access Amazon S3 bucket



Among these, the **AWS S3 bucket** was ideal for our needs. It allowed us to access the entire OpenAQ dataset efficiently and flexibly.
We developed a set of Python scripts designed to fetch the data systematically—each script retrieving one year's worth of data. This modular approach enabled us to distribute tasks across multiple small-sized EC2 instances simultaneously, optimizing resource utilization and significantly speeding up the process.
This practical experience also served as a hands-on exercise in parallel computing. By leveraging distributed tasks and multiple CPUs, we gained valuable insights into handling large-scale data ingestion effectively.



Challenge of Millions of Files: After completing the full ingestion—taking several days to finish—we ended up with millions upon millions of individual data files stored on AWS S3. The number of files was so large that AWS's standard interface spent hours to calculate the total size of our dataset.



Content:
- Total data size: ~19.3 GB
- Total number of objects: ~28 million!

While each individual file was quite small and efficiently compressed in CSV.GZ format, the sheer number was drastically reducing the performance of our Spark jobs. To overcome this challenge, we decided to convert the entire dataset into **Parquet format**, which significantly improves query speed, optimizes storage, and streamlines big data analysis.

3. Turkish National Air Quality Monitoring System (UHKiA)

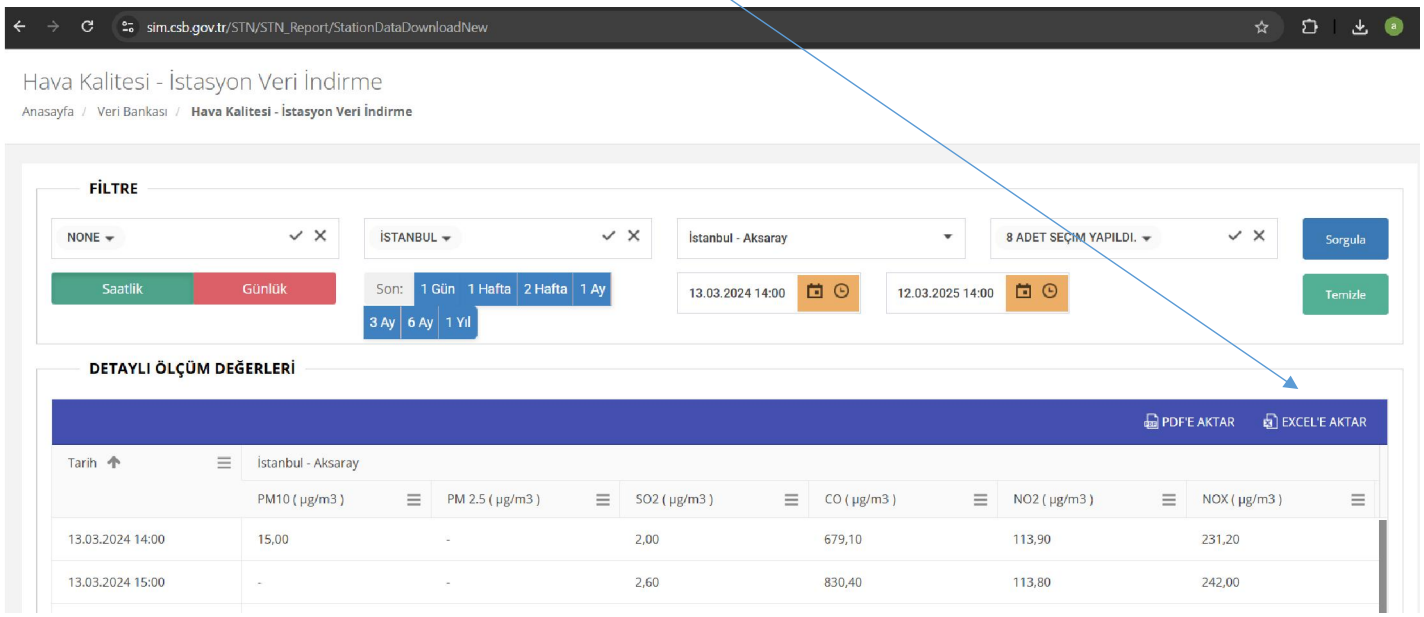
The **Ulusal Hava Kalitesi İzleme Ağı (UHKiA)**, managed by the **T.C. Çevre, Şehircilik ve İklim Değişikliği Bakanlığı**, is a nationwide initiative launched in 2005 to systematically monitor Turkey's air quality across all provinces. The system supports environmental policy-making by providing reliable, continuous air quality data through a growing network of monitoring stations.
UHKiA provides public access to multiple environmental monitoring modules through its official platform (sim.csb.gov.tr)

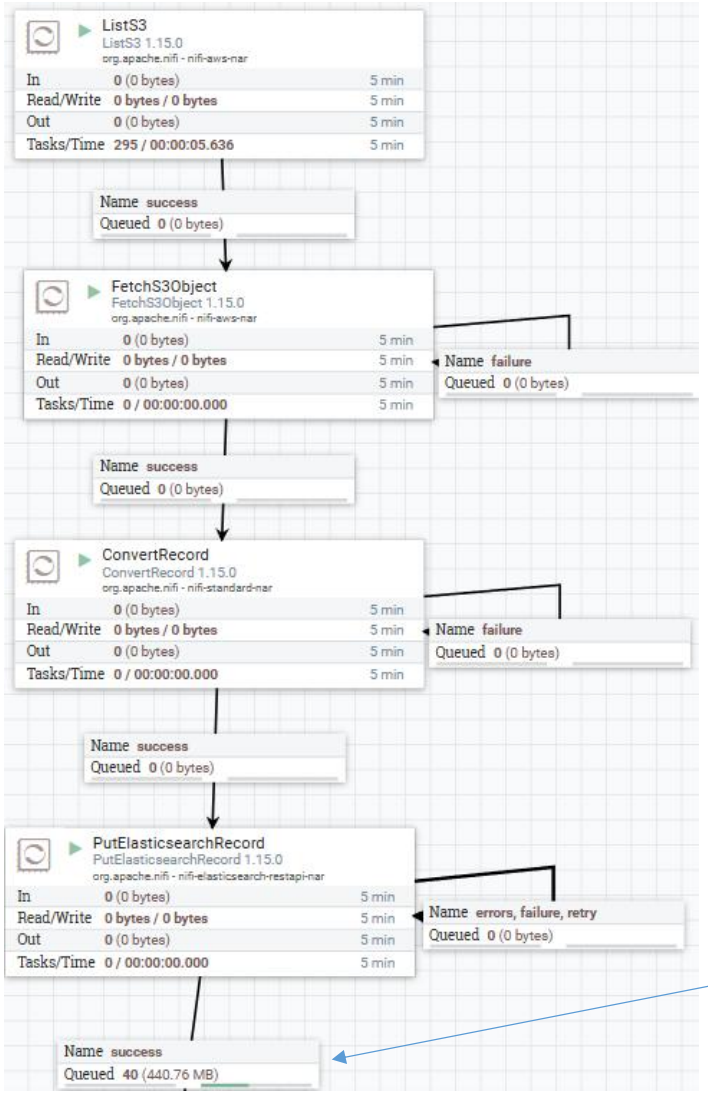
SİM Veri Bankası

For this project, we specifically used the **SİM Veri Bankası** module, which aggregates historical and real-time measurement data from **all air quality monitoring stations across Turkey**.
To build our Air Quality Data Platform, we manually downloaded historical data for Istanbul's air quality, dating back to the earliest recorded data. We focused on downloading Excel files (.xlsx) that contain time-stamped measurements for each station.

Data Conversion Process

Once we obtained the historical data, the next step was to convert these Excel files into CSV format for easier processing. We used a Python script to batch convert each .xlsx file into a .csv file. After conversion, these files were uploaded to an AWS S3 bucket, making them accessible to our data pipeline.





NiFi Configuration

Using Apache NiFi, we set up a data pipeline to automate the processing and ingestion of the Istanbul's historical air quality data from the S3 bucket into Elasticsearch. The key components included:

- **ListS3**: To list files from the S3 bucket.
- **FetchS3Object**: To retrieve the CSV files.
- **ConvertRecord**: To convert CSV data into JSON format using a CSVReader and a JsonRecordSetWriter.
- **PutElasticsearchRecord**: To push the processed JSON data into Elasticsearch.

With data successfully indexed into Elasticsearch, we configured Kibana to create visualizations.

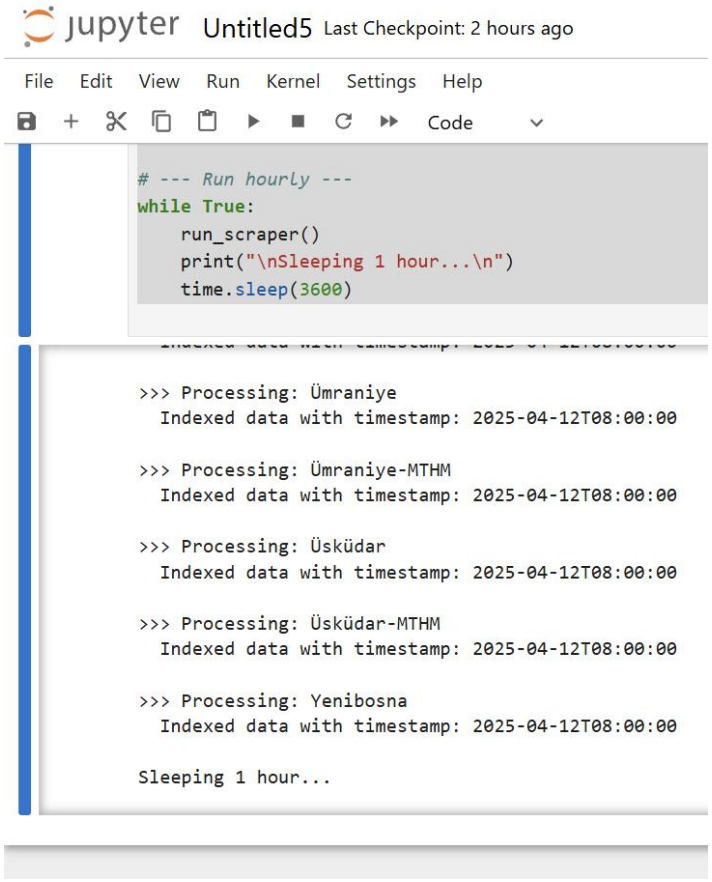
Real-Time Air Quality Data Integration

Overview

We wanted to incorporate real-time data into our data platform. We initially tried using Waqi, but after facing IP blocking issues, we moved on to the real data source: HavaİzleMe. Since HavaİzleMe doesn't provide an API and blocks exterior IPs outside Turkey, we implemented the solution locally on our computers, scraping the data directly from the website.

Implementation

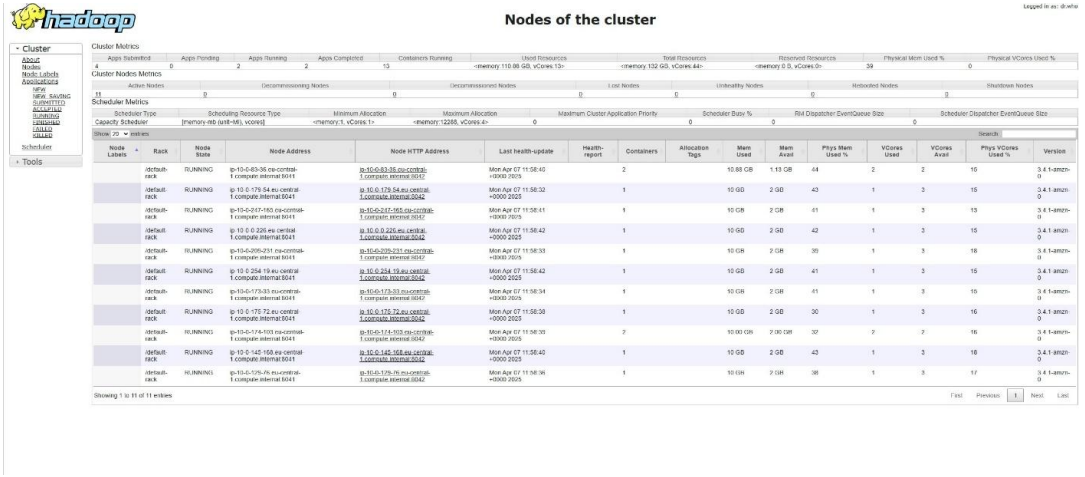
To scrape real-time data from HavaİzleMe, we developed an automated process using Python. The script runs at one-hour intervals, collecting data for various pollutants. If a value is missing, the script uses the previous hour's data. The processed data is then sent to Elasticsearch, ensuring the latest information is always available for visualization in Kibana.



Converting Data to Parquet Format

- Amount of the csv files (28 million)
- Memory Overload Warning
- Extracting all file paths with EC2 machines by running a python script.
- EMR - 1 core vs 10 core machines and machine configuration
- Feeding the parquet spark script with the paths and using partitioning

s3_path	location_id
s3://horizon-bigdataingestion/openaq-data/locationid=1011736/year=2024/month=04/location-1011736-20240424.csv.gz	1011736
s3://horizon-bigdataingestion/openaq-data/locationid=1011736/year=2024/month=04/location-1011736-20240426.csv.gz	1011736
s3://horizon-bigdataingestion/openaq-data/locationid=1011736/year=2024/month=04/location-1011736-20240427.csv.gz	1011736
s3://horizon-bigdataingestion/openaq-data/locationid=1011736/year=2024/month=04/location-1011736-20240428.csv.gz	1011736
s3://horizon-bigdataingestion/openaq-data/locationid=1011736/year=2024/month=04/location-1011736-20240429.csv.gz	1011736
s3://horizon-bigdataingestion/openaq-data/locationid=1011736/year=2024/month=04/location-1011736-20240430.csv.gz	1011736
s3://horizon-bigdataingestion/openaq-data/locationid=1011736/year=2024/month=05/location-1011736-20240501.csv.gz	1011736
s3://horizon-bigdataingestion/openaq-data/locationid=1011736/year=2024/month=05/location-1011736-20240502.csv.gz	1011736



Spark Job – Getting Ready For Visualization

İSTANBUL TEKNİK ÜNİVERSİTESİ

- Struggled while reading the parquet formats thats why parquet formats are merged
- Cleaning the raw data (mistaken measurements, missing values)
- Defining the schema of the data
- Extracting new columns like year, month, day
- Merging latitude and longitude data with geonames data for Country, city_name, population columns
- Converting to JSON

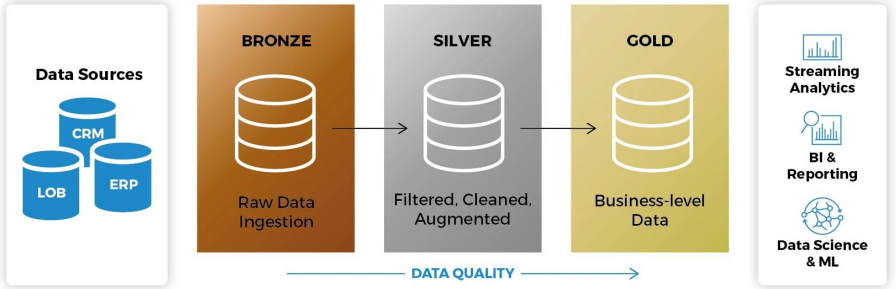
```
%pyspark
df_openaq_conv.filter("value < 0").show()
```

location_id	sensors_id	location	datetime	lat	lon	parameter	units	value	year	month	day
100	4275786	Badhoevedorp-Slot...	2024-01-11 12:00:00	52.334	4.77401	no	µg/m³	-999.0	2024	1	11
100	4275786	Badhoevedorp-Slot...	2024-01-28 18:00:00	52.334	4.77401	no	µg/m³	-999.0	2024	1	28
100	4231	Badhoevedorp-Slot...	2024-02-17 12:00:00	52.334	4.77401	pm10	µg/m³	-995.0	2024	2	17
100	4231	Badhoevedorp-Slot...	2024-02-24 18:00:00	52.334	4.77401	pm10	µg/m³	-995.0	2024	2	24
100	4147	Badhoevedorp-Slot...	2024-01-28 21:00:00	52.334	4.77401	pm25	µg/m³	-999.0	2024	1	28
100	4275786	Badhoevedorp-Slot...	2024-02-10 21:00:00	52.334	4.77401	no	µg/m³	-999.0	2024	2	10
100	4147	Badhoevedorp-Slot...	2024-02-25 16:00:00	52.334	4.77401	pm25	µg/m³	-999.0	2024	2	25
100	4147	Badhoevedorp-Slot...	2024-01-21 02:00:00	52.334	4.77401	pm25	µg/m³	-999.0	2024	1	21
100	4147	Badhoevedorp-Slot...	2024-02-10 00:00:00	52.334	4.77401	pm25	µg/m³	-999.0	2024	2	10
100	162	Badhoevedorp-Slot...	2024-01-28 10:00:00	52.334	4.77401	co	µg/m³	-999.0	2024	1	28
100	4275786	Badhoevedorp-Slot...	2024-01-24 21:00:00	52.334	4.77401	no	µg/m³	-999.0	2024	1	24
100	162	Badhoevedorp-Slot...	2024-01-25 10:00:00	52.334	4.77401	co	µg/m³	-999.0	2024	1	25
100	162	Badhoevedorp-Slot...	2024-01-13 13:00:00	52.334	4.77401	co	µg/m³	-999.0	2024	1	13
100	162	Badhoevedorp-Slot...	2024-02-10 08:00:00	52.334	4.77401	co	µg/m³	-999.0	2024	2	10

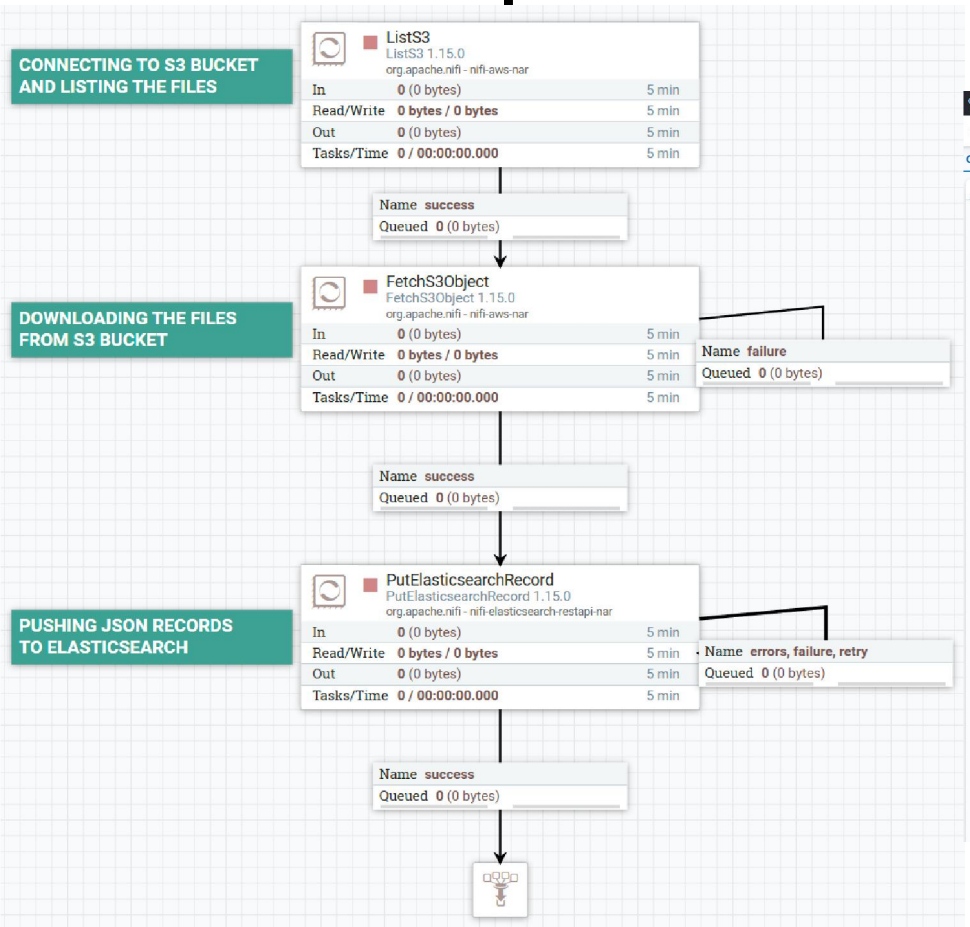
Took 7 sec. Last updated by anonymous at April 09 2025, 3:33:25 PM.

location_id	sensors_id	country	city_name	population	parameter	units	value	datetime	year	month	day
100	162	The Netherlands	Badhoevedorp	0	co	µg/m³	0.4812	2024-03-09 11:00:00	2024	3	9
100	162	The Netherlands	Badhoevedorp	0	co	µg/m³	0.4393	2024-03-10 23:00:00	2024	3	10
100	162	The Netherlands	Badhoevedorp	0	co	µg/m³	0.6432	2024-03-11 17:00:00	2024	3	11
100	162	The Netherlands	Badhoevedorp	0	co	µg/m³	0.528	2024-03-11 18:00:00	2024	3	11
100	162	The Netherlands	Badhoevedorp	0	co	µg/m³	0.3348	2024-03-17 01:00:00	2024	3	17
100	162	The Netherlands	Badhoevedorp	0	co	µg/m³	0.3367	2024-03-17 12:00:00	2024	3	17
100	162	The Netherlands	Badhoevedorp	0	co	µg/m³	0.3258	2024-03-17 14:00:00	2024	3	17
100	162	The Netherlands	Badhoevedorp	0	co	µg/m³	0.2565	2024-03-19 15:00:00	2024	3	19
100	162	The Netherlands	Badhoevedorp	0	co	µg/m³	0.2758	2024-03-19 17:00:00	2024	3	19
100	162	The Netherlands	Badhoevedorp	0	co	µg/m³	0.2522	2024-03-22 03:00:00	2024	3	22
100	162	The Netherlands	Badhoevedorp	0	co	µg/m³	0.2651	2024-03-22 08:00:00	2024	3	22
100	162	The Netherlands	Badhoevedorp	0	co	µg/m³	0.2839	2024-03-23 11:00:00	2024	3	23
100	162	The Netherlands	Badhoevedorp	0	co	µg/m³	0.2433	2024-03-24 03:00:00	2024	3	24
100	162	The Netherlands	Badhoevedorp	0	co	µg/m³	0.247	2024-03-24 06:00:00	2024	3	24

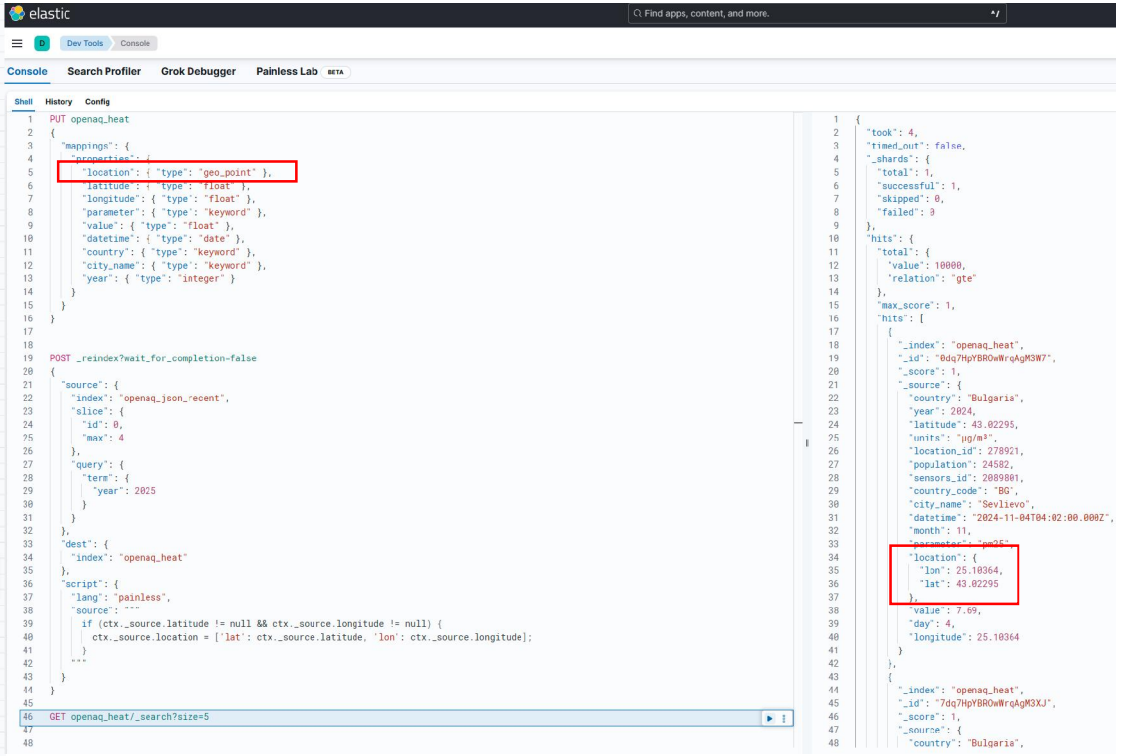
Took 7 sec. Last updated by anonymous at April 09 2025, 3:26:59 PM.



Nifi Pipeline

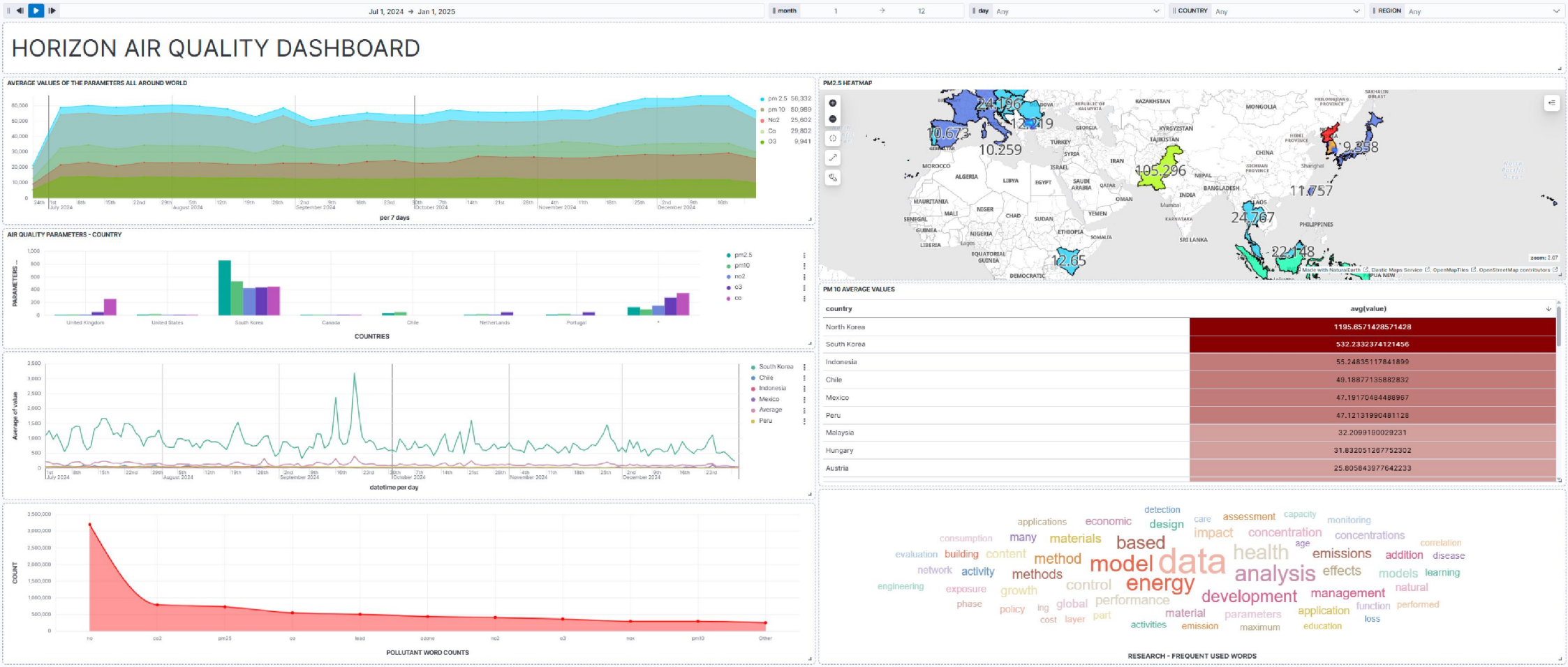


Kibana – Creating new index (Geopoint transformation)



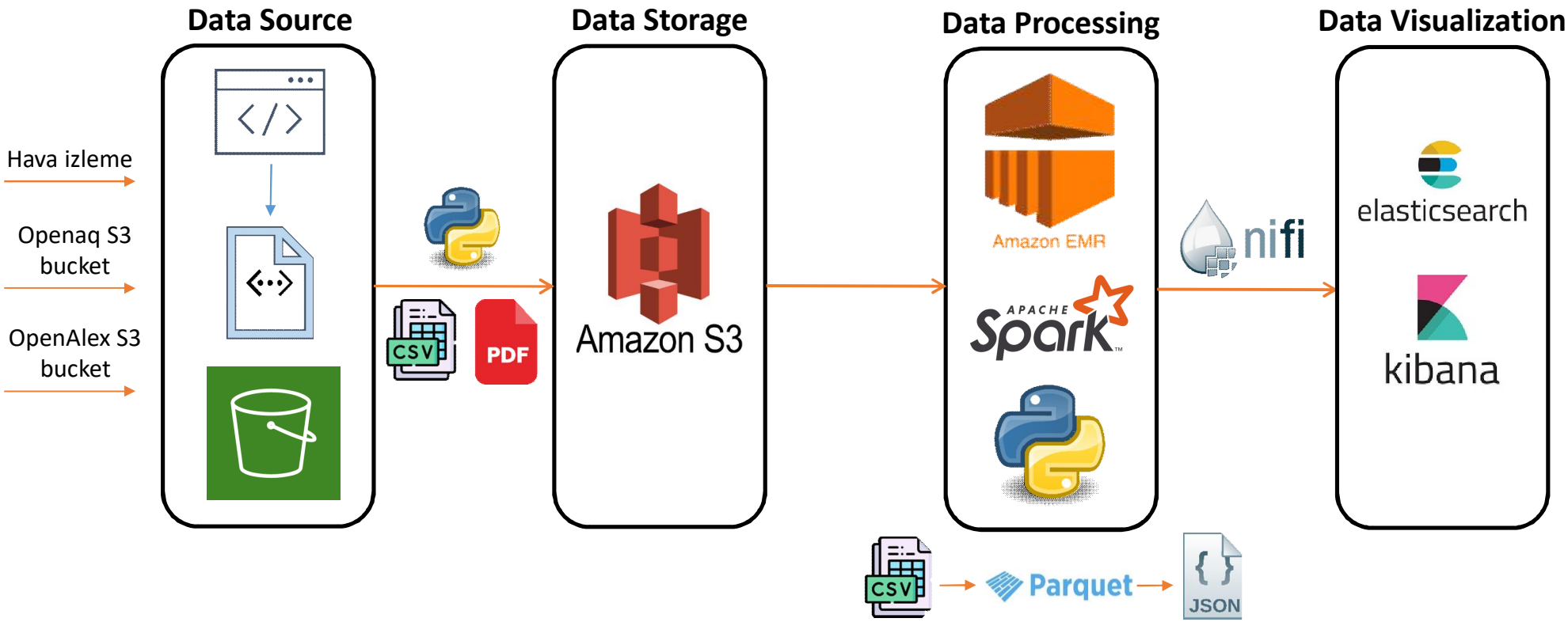
KIBANA DASHBOARD

İSTANBUL TEKNİK ÜNİVERSİTESİ



Data Architecture

İSTANBUL TEKNİK ÜNİVERSİTESİ



Future Goals of the Air Quality Data Platform

1.Automation of Data Source Ingestion

Develop scheduled, fully automated pipelines for integrating data from all sources, minimizing manual processing and improving consistency.

2.Machine Learning Integration

Implement machine learning algorithms to analyze both structured and unstructured data. Applications include forecasting air quality trends, anomaly detection, and predictive modeling at the station level.

3.Kafka-Based Real-Time Pipeline

Introduce Apache Kafka as a real-time data streaming layer for the HavaİzIeme scraper. This will enable better scalability, decoupling of components, and support for multiple downstream consumers such as Elasticsearch, dashboards, and archival storage.

4.Nationwide Coverage of HavaİzIeme Data

Expand the current Istanbul-focused scraping framework to cover all available monitoring stations across Turkey for more comprehensive air quality insights.

5.Scalable Spark Processing of OpenAQ Data

Complete and optimize the Spark-based ETL processes for OpenAQ's large-scale dataset. Future iterations will focus on handling full-year data more efficiently using distributed compute resources.

6.Cloud Migration of HavaİzIeme Pipeline

Explore methods to migrate the currently local HavaİzIeme scraping pipeline to the cloud, potentially through VPN tunneling or proxy routing to overcome geographic access restrictions.

7.Incorporation of GDELT Data

Integrate the Global Database of Events, Language, and Tone (GDELT) as a complementary dataset. This may provide valuable correlations between environmental metrics and geopolitical, media, or social events.

Thank You!