Idea Proposal

Air Quality Trends and
Thermal Power Correlation in Korea
(2003–2024)

Team 1
21102061 Hwang Hyunmin
21102052 Lee Jeongyun
23102020 Lee Sodam
23102025 Lee Haneol

Contents

Basic Idea

Problem

Sustained air quality degradation since 2003, driven by economic growth and increased energy demand.

Hypothesis:
The rise in national thermal power generation has negatively impacted air quality.

Goal

- Quantitatively analyze long-term trends linking energy demand and air quality.
- Systematically explore the correlation between thermal power output and pollutant concentration.

Establishment of a **"Self-updating Analytical Pipeline"** by automating data collection, loading, and analysis on a monthly basis.

System overview and architecture

Flow

Data Source

Storage & Management

Processing

Analysis

Output

System overview and architecture

Data Source

Why automated collection?

- **Regular Data Updates** - Automation ensures new data is collected on schedule without manual effort.
- **Consistency & Accuracy** - Uses the same extraction process every time → prevents human error. Guarantees consistent file format.
- **Efficiency** - Saves time by removing repetitive manual downloads.
- **Reliability** - System runs even without human intervention.
- **Integrated Pipeline** - Automated flow: Web Crawling → Hadoop HDFS → Hive → Python Analysis

System overview and architecture

Storage & Management

- **Why Hadoop?**

-Designed to **handle multi-year, large-scale air quality data efficiently**

-Provides distributed storage and automatic replication **for reliability**

-Enables **parallel access and high throughput** for data processing

-Selected to **support scalable, fault-tolerant storage** within our ETL pipeline

- **Why Hive?**

-**Integrated smoothly with Spark and HDFS** for batch data processing

-Enabled Parquet conversion and partitioning to optimize query speed

-Supported automated **ETL scheduling** for a self-updating pipeline

-Chosen over Impala as it **fits monthly batch analysis** better than real-time querying

System overview and architecture

Processing

Why Spark?

- **Spark** is an essential distributed processing framework for handling **massive datasets** like 21 years of hourly data
- Used to avoid the **slow speeds** and potential **memory errors** (OOM) of single-server Python environments
- Performs complex aggregation operations like groupBy and avg **dozens of times faster** by distributing data across multiple nodes

→ **Pre-processes the air quality data into monthly averages efficiently!**

System overview and architecture

python™

Analysis

Why Python?

- To perform flexible data analysis, correlation, and visualization.
- To perform flexible data analysis, correlation, and visualization.
- We import Spark's results (CSV or Parquet) into Python to compute correlation coefficients like Pearson and Spearman, and to visualize results with heatmaps and line charts.
- In short, Python is ideal for **in-depth statistical analysis and visualization after distributed processing**.

System overview and architecture

Streamlit

Output

Why Streamlit?

- We wanted to make our analysis results easy to explore and share on the web.
- So, we built an **interactive dashboard** instead of using static charts.
- Streamlit, **a Python-based framework**, lets us create such dashboards quickly without any HTML or CSS, **supporting real-time visualization** and data filtering.

System overview and architecture

Flow:
Data Source → Storage & Management → Processing → Analysis → Output

① Data Source Layer

- ### Data Composition

No.	Source	Main Attributes	Period	Format	Purpose
①	**Air Quality Data (AirKorea, Ministry of Environment)**	Region, Station Code, Timestamp, SO_2, NO_2, O_3, CO, PM10, PM 2.5, Address	2003–2024	CSV (monthly files)	Analyze hourly air pollution trends nationwide
②	**Power Generation Data (KEPCO)**	Year, Region, Power Generation (MWh)	2003–2024	CSV (annual statistics)	Examine thermal power generation and electricity usage patterns

System overview and architecture

Flow:
Data Source → Storage & Management → Processing → Analysis → Output

① Data Source Layer

Automated Data Collection:
– Collect monthly finalized air quality and power generation data using Linux cron scheduler and Python BeautifulSoup (web crawling).
– Automatically executed on the 1st day of each month to retrieve data from the previous month.

구분 / 월별	기력 Steam				
	무연탄 Anthracite coal	유연탄 Bituminous coal	중유 Heavy oil	LNG	계 Total
1	258,157	14,317,161	–	20,781	14,596,098
2	138,412	11,818,512	–	13,200	11,970,123
3	117,616	10,026,705	–	33,043	10,177,364
4	122,285	9,404,183	–	13,670	9,540,138
5	183,507	8,869,192	–	26,641	9,079,339
6	195,036	10,789,182	–	85,411	11,069,629
7	178,624	14,185,887	–	85,437	14,449,949
8	240,716	15,918,481	–	161,128	16,320,325
9	196,978	12,441,738	–	68,539	12,707,255
10	25,838	9,676,802	–	64,663	9,767,303
11	123,402	9,574,087	–	11,940	9,709,429
12	210,182	11,792,530	–	1,973	12,004,685

	A	B	C	D	E	F	G	H	I	J	K
1	지역	측정소명	측정소코드	측정일시	SO2	CO	O3	NO2	PM10	주소	
2	서울	중구	111121	2013010101	0.006	1.1	0.003	0.061	30	서울 중구 서소문동	
3	서울	중구	111121	2013010102	0.006	1.1	0.003	0.058	42	서울 중구 서소문동	
4	서울	중구	111121	2013010103	0.006	0.9	0.003	0.051	46	서울 중구 서소문동	
5	서울	중구	111121	2013010104	0.006	0.9	0.004	0.046	30	서울 중구 서소문동	
6	서울	중구	111121	2013010105	0.005	0.8	0.005	0.039	25	서울 중구 서소문동	
7	서울	중구	111121	2013010106	0.005	0.9	0.004	0.041	34	서울 중구 서소문동	

System overview and architecture

Flow:
Data Source → Storage & Management → Processing → Analysis → Output

Web Crawling:

Implementation

- Developed a Python-based web crawler using the requests library.
- The script sends a GET request to the official AirKorea endpoint.
- Downloads daily measurement data as an Excel file (.xls) for each monitoring station.

Automation

- Scheduled via Linux cron job.
- Runs automatically on the 1st day of each month.
- Retrieves the previous month’s data without manual intervention.

System overview and architecture

Flow:
Data Source → Storage & Management → Processing → Analysis → Output

② Data Storage & Management Layer

Hadoop HDFS

- **Path**: /user/airquality/year=YYYY/month=MM/
- **Function**: Stores daily air quality data in raw CSV format before preprocessing.
- **Feature**: Splits files into blocks and distributes them across multiple DataNodes for parallel processing and reliability.

Hive (Data Warehouse)

- **Role**: Enables SQL-like querying on data stored in HDFS / CSV → Parquet conversion
- **Table Structure:**
  - **External table**: air_quality (raw)
  - **Partitions**: year, month
  - **Main columns**: region, timestamp, so2, no2, pm10, pm25, etc.

System overview and architecture

Flow:
Data Source → Storage & Management → Processing → Analysis → Output

③ Data Processing Layer

Spark

- Calculate the monthly average for 21 years of national hourly air quality data

1) **Data Preprocessing & Loading**:
– Load hourly CSV files from HDFS into Spark and normalize column names (KR→EN).
– Cast data types and fill missing values with mean.
– Save cleaned data as Parquet and register in Hive.

2) Time Attribute Extraction:
– Extract Year and Month from the timestamp column to use as keys for correlation analysis.

System overview and architecture

Flow:
Data Source → Storage & Management → Processing → Analysis → Output

③ Data Processing Layer

Spark

- Calculate the monthly average for 21 years of national hourly air quality data

3) **Monthly Aggregation**:
– Perform groupBy(year, month) and apply avg() for each pollutant to calculate monthly national averages, aligning the time scale to enable equivalent comparison with power statistics.

4) **Data Extraction:**
– Export the aggregated results as CSV files for subsequent Python-based correlation and visualization.

System overview and architecture

Flow:
Data Source → Storage & Management → Processing → Analysis → Output

④ Data Analysis & Visualization Layer

Python (pandas, scipy, seaborn)

- Merge Spark output and power generation data into a unified dataset.
- Perform correlation (Pearson, Spearman), trend, and seasonality analyses to detect time-series patterns.
- Visualize results with heatmaps and line charts.

Streamlit (Output visualization)

- Develop an interactive web dashboard to dynamically visualize analysis results.
- Enable users to explore timely trends, correlations, and download filtered data in real time.

System overview and architecture

Flow:
Data Source → Storage & Management → Processing → Analysis → Output

⑤ Output Layer

1.Correlation Results

- **Thermal Power Generation ↔ Air Quality Indicators (PM10, NO_2, SO_2):**

Calculate correlation coefficients to examine how power generation intensity affects air pollution levels.

- **Temporal Correlation Changes:**

Visualize how the relationship between power generation and air pollutants evolves over time (2003–2024).

System overview and architecture

Flow:
Data Source → Storage & Management → Processing → Analysis → Output

⑤ Output Layer

2.Trend Visualization

- **Time-Series Trends:**

Display nationwide trends in PM 10 and NO_2 concentrations over 21 years.

- **Monthly Generation vs. Pollution Curves:**

Plot comparative graphs showing monthly generation output and pollutant fluctuations.

- **Automated Monitoring Insight:**

Demonstrate the potential of a self-updating analytical pipeline that continuously collects, processes, and visualizes data for real-time environmental monitoring.

System overview and architecture

Flow:
Data Source → Storage & Management → Processing → Analysis → Output

Extract

Data Source 1:
Air Quality Data
(AirKorea)

Data Source 2:
Power Generation Data
(KEPCO)

Transform

Transformation Engine:
Apache Spark

Transformation Logic:
① Aggregate of air quality data
② Prepare for combining with power generation data

Load

Target:
Python & Streamlit

System overview and architecture

Cluster Configuration: 3 Node Cluster

One Master Node:

- NameNode in HDFS
- Cluster Resource Management
- DB Schema Management (Hive Metastore)
- Spark Task Distribution

Two DataNodes:

- Storage of Data Blocks
- Spark Job Execution

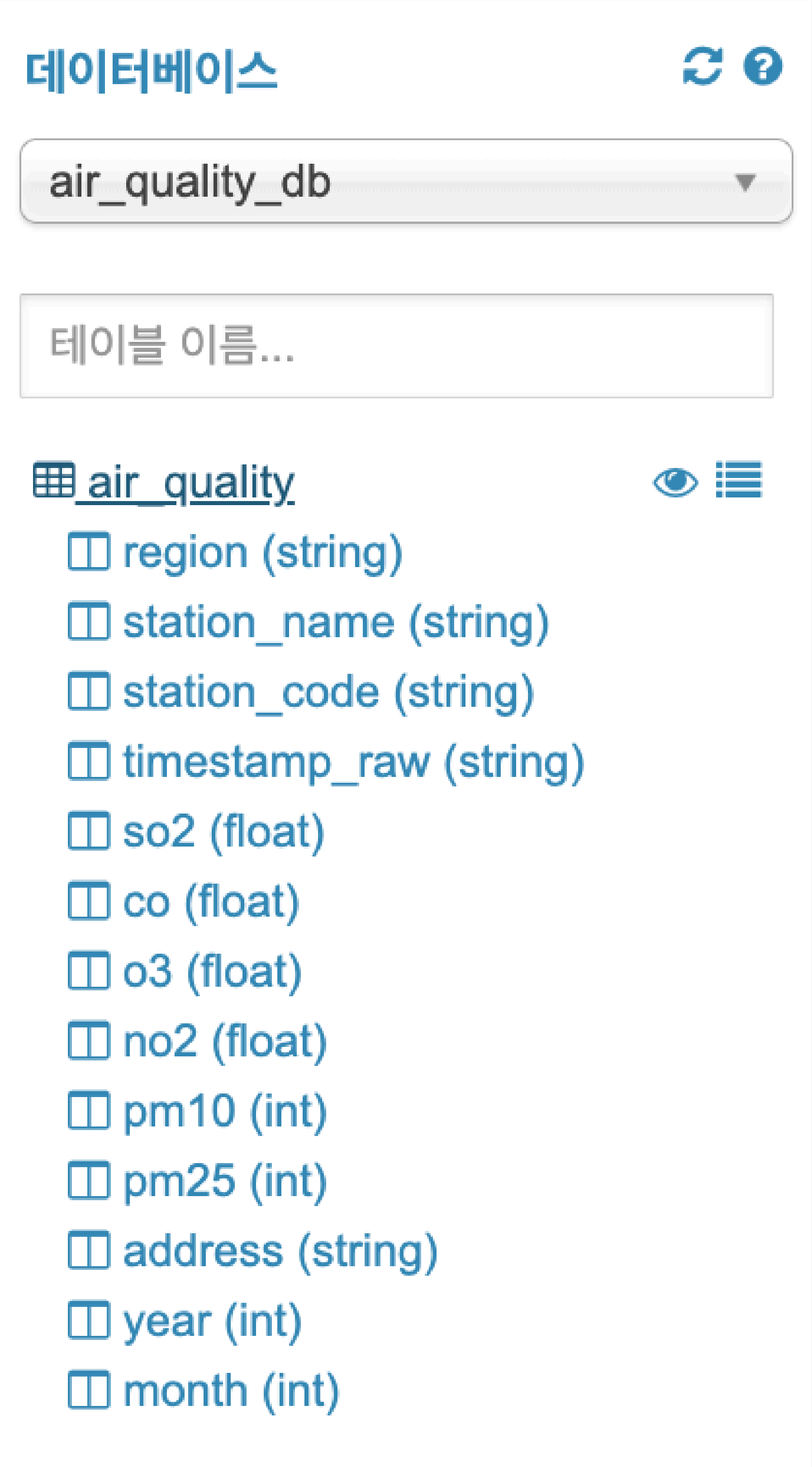
One Client:

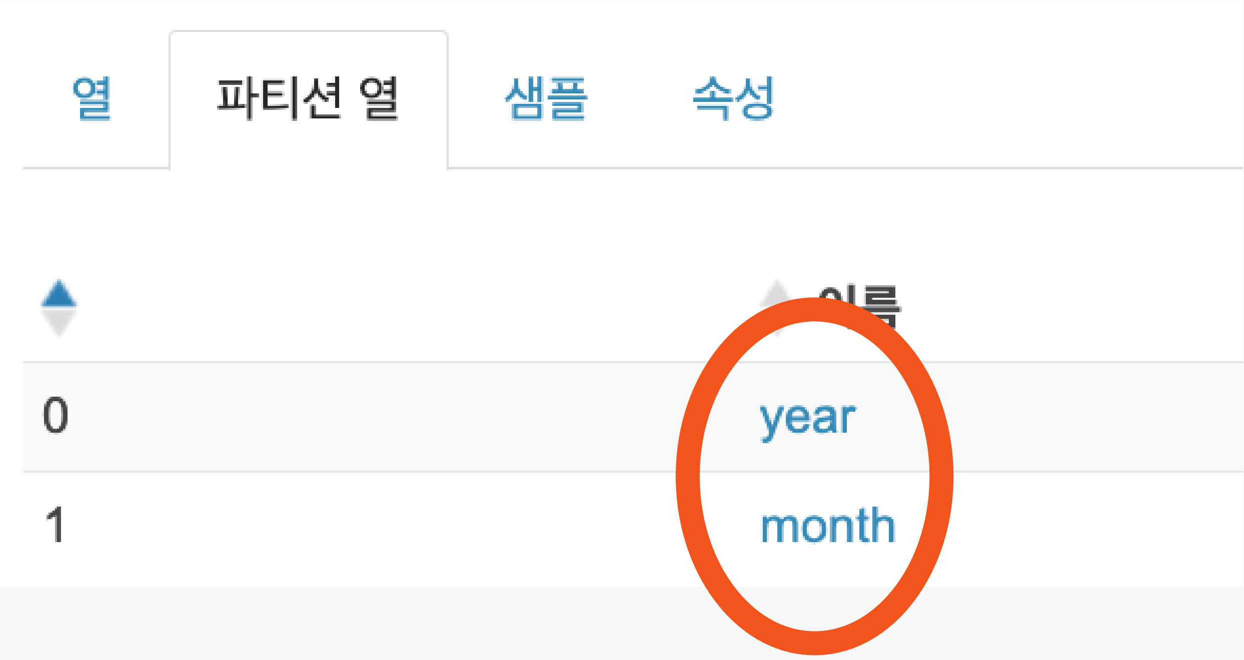
- Automated Data Collection
- Python-based Analysis

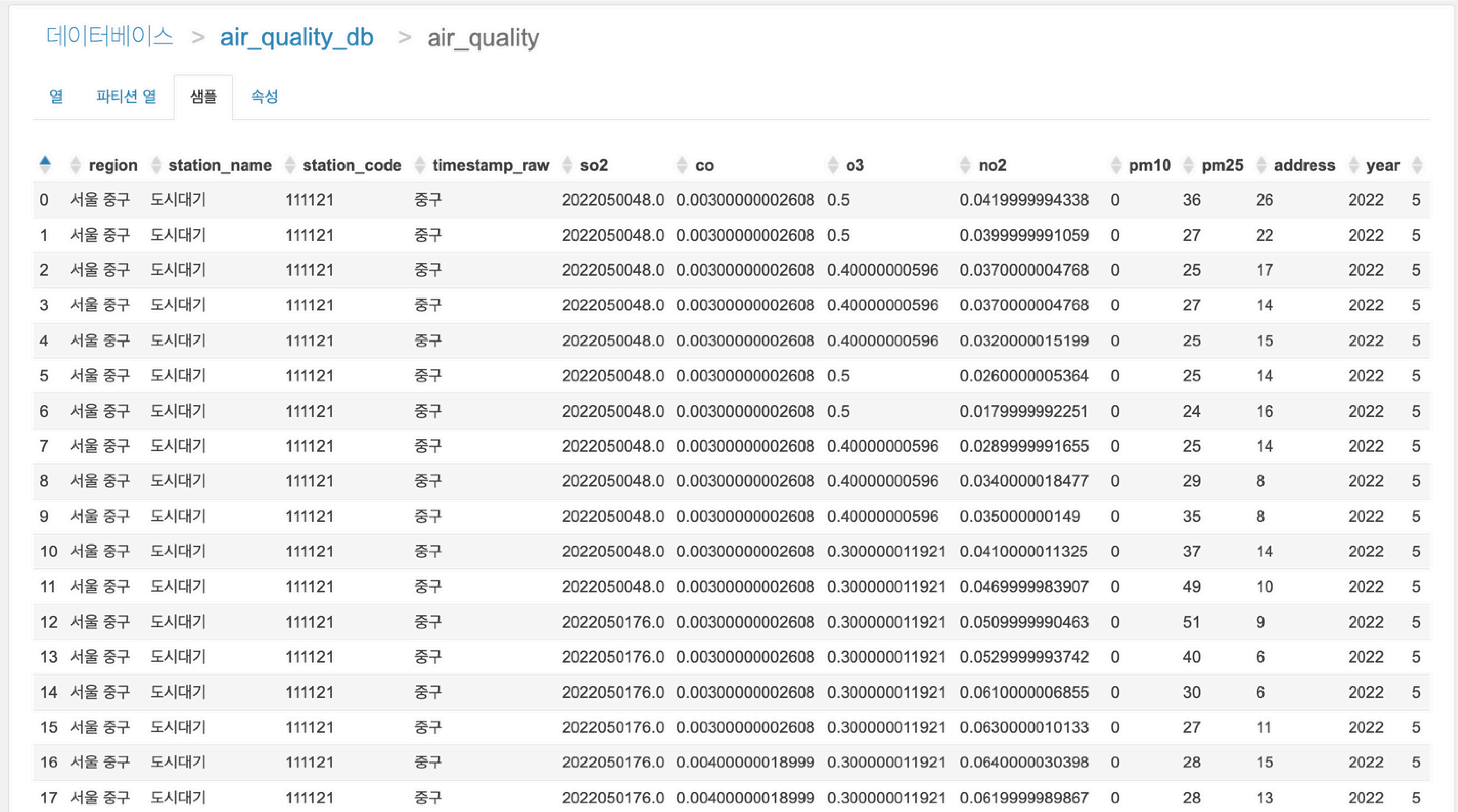
- **Implementation: Docker based VM:**

Each node runs independently in a separate container within the same Docker network, enabling efficient inter-node communication and easy scalability.

Intermediate results





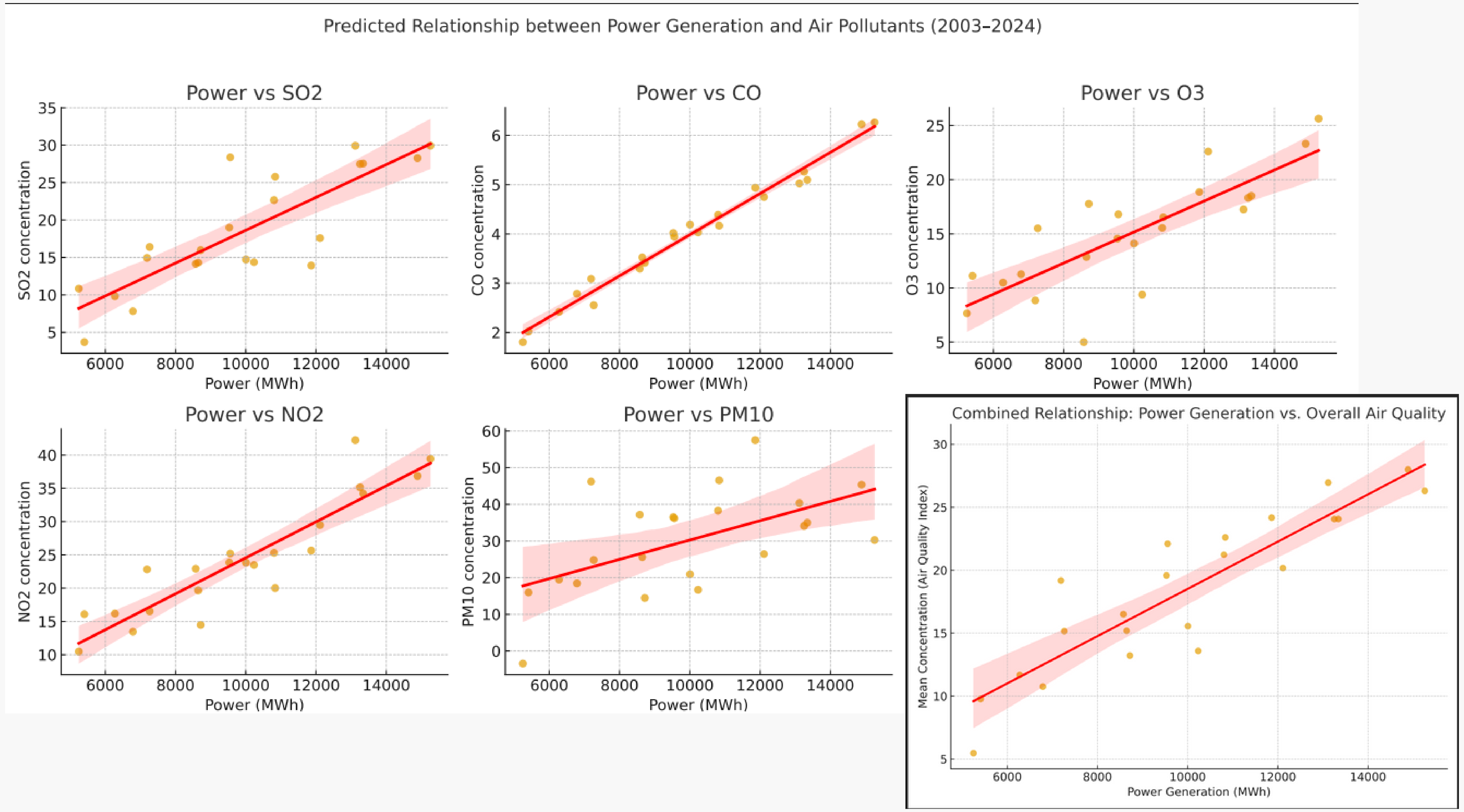

데이터베이스 > air_quality_db > air_quality

열 파티션 열 샘플 속성

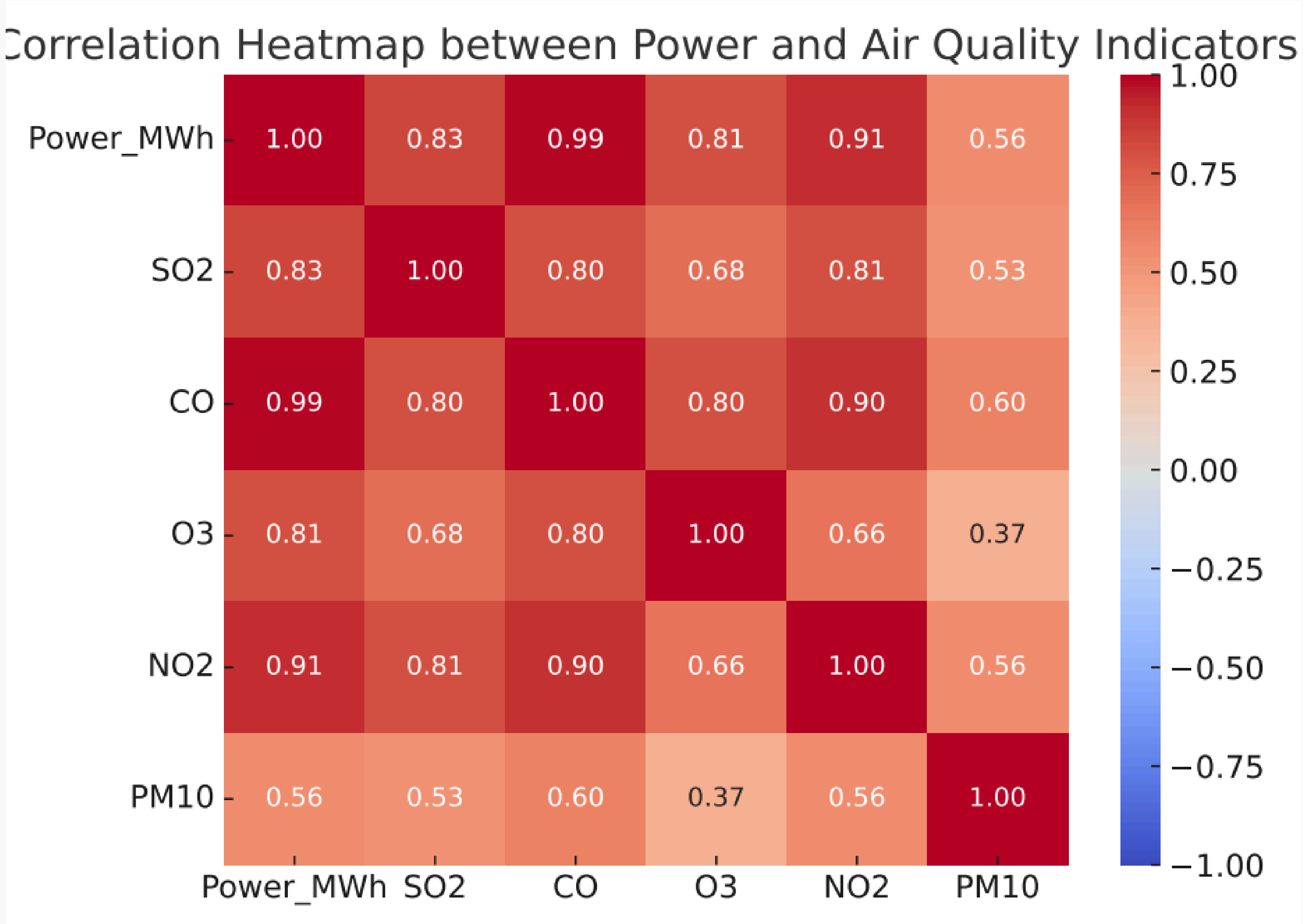
	region	station_name	station_code	timestamp_raw	so2	co	o3	no2	pm10	pm25	address	year	
0	서울 중구	도시대기	111121	중구	2022050048.0	0.00300000002608	0.5	0.0419999994338	0	36	26	2022	5
1	서울 중구	도시대기	111121	중구	2022050048.0	0.00300000002608	0.5	0.0399999991059	0	27	22	2022	5
2	서울 중구	도시대기	111121	중구	2022050048.0	0.00300000002608	0.40000000596	0.0370000004768	0	25	17	2022	5
3	서울 중구	도시대기	111121	중구	2022050048.0	0.00300000002608	0.40000000596	0.0370000004768	0	27	14	2022	5
4	서울 중구	도시대기	111121	중구	2022050048.0	0.00300000002608	0.40000000596	0.0320000015199	0	25	15	2022	5
5	서울 중구	도시대기	111121	중구	2022050048.0	0.00300000002608	0.5	0.0260000005364	0	25	14	2022	5
6	서울 중구	도시대기	111121	중구	2022050048.0	0.00300000002608	0.5	0.0179999992251	0	24	16	2022	5
7	서울 중구	도시대기	111121	중구	2022050048.0	0.00300000002608	0.40000000596	0.0289999991655	0	25	14	2022	5
8	서울 중구	도시대기	111121	중구	2022050048.0	0.00300000002608	0.40000000596	0.0340000018477	0	29	8	2022	5
9	서울 중구	도시대기	111121	중구	2022050048.0	0.00300000002608	0.40000000596	0.035000000149	0	35	8	2022	5
10	서울 중구	도시대기	111121	중구	2022050048.0	0.00300000002608	0.300000011921	0.0410000011325	0	37	14	2022	5
11	서울 중구	도시대기	111121	중구	2022050048.0	0.00300000002608	0.300000011921	0.0469999983907	0	49	10	2022	5
12	서울 중구	도시대기	111121	중구	2022050176.0	0.00300000002608	0.300000011921	0.0509999990463	0	51	9	2022	5
13	서울 중구	도시대기	111121	중구	2022050176.0	0.00300000002608	0.300000011921	0.0529999993742	0	40	6	2022	5
14	서울 중구	도시대기	111121	중구	2022050176.0	0.00300000002608	0.300000011921	0.0610000006855	0	30	6	2022	5
15	서울 중구	도시대기	111121	중구	2022050176.0	0.00300000002608	0.300000011921	0.0630000010133	0	27	11	2022	5
16	서울 중구	도시대기	111121	중구	2022050176.0	0.00400000018999	0.300000011921	0.0640000030398	0	28	15	2022	5
17	서울 중구	도시대기	111121	중구	2022050176.0	0.00400000018999	0.300000011921	0.0619999989867	0	28	13	2022	5

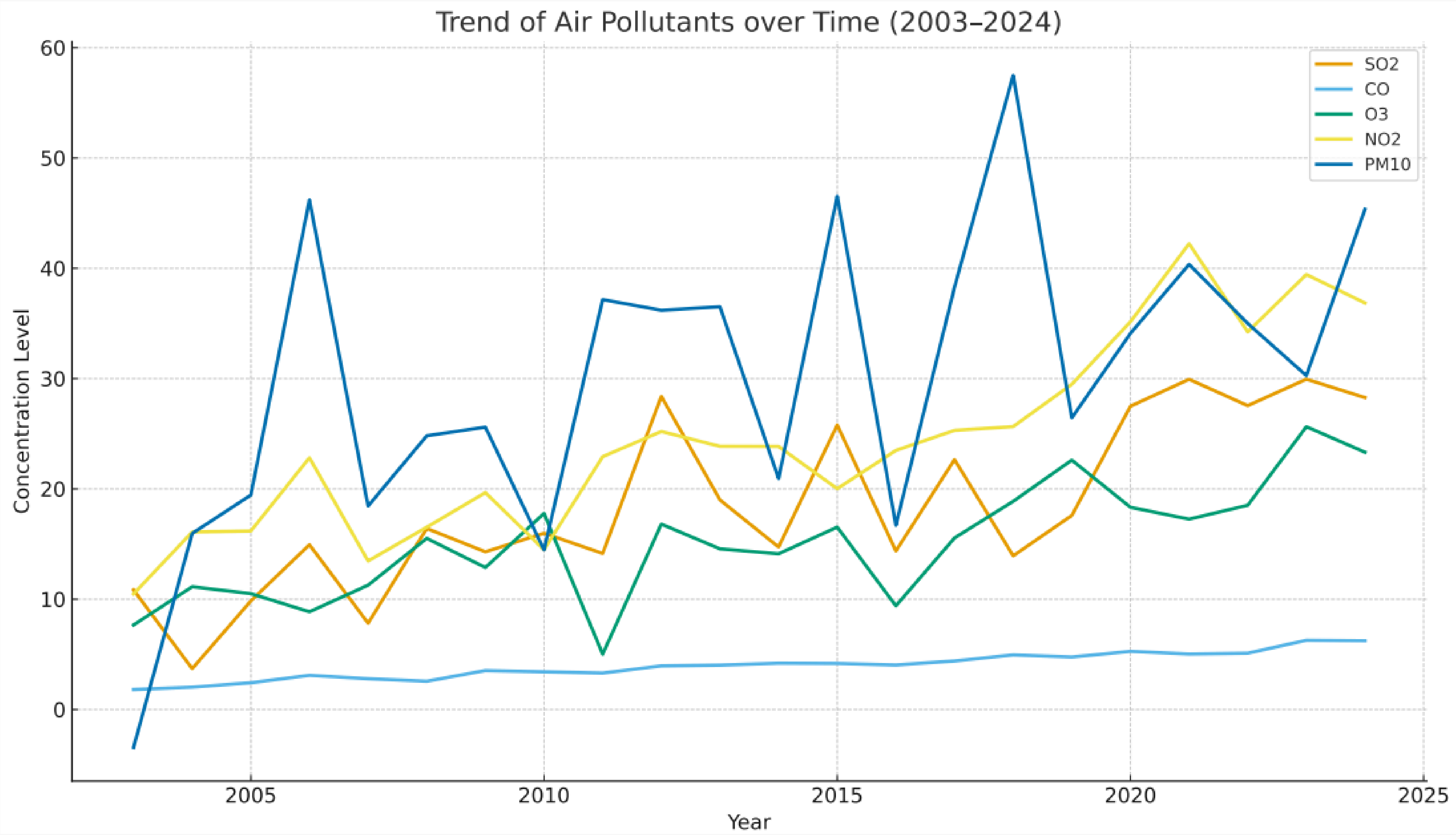
Expected results

- Increased thermal power generation shows strong positive correlations with SO_2, CO, NO_2, and PM10.
- Seasonal patterns indicate higher pollutant levels during high energy-demand periods.
- Results support the hypothesis that rising power output has negatively impacted air quality.



Expected results





Project plan

Main Task	Sub Task	~W6	W7	W8	W9	W10	W11	W12	W13	W14 (12/1)
1. Project Planning	Project Schedule Planning									
	Role Assignment and Documentation									
2. Data Source Layer	AirKorea / KEPCO Data Verification									
3. Data Storage & Management	HDFS Configuration and Hive Table Setup									
	CSV Upload and Midterm Report Compilation									
4. Data Processing (Spark)	Spark Environment Setup and Monthly Average Calculation									
5. Analysis & Visualization	Python Environment Setup									
	Correlation Analysis and Streamlit Prototype									
6. Output Layer	Final Visualization and Report Writing									
7. Presentation	PPT Creation and Presentation Recording									

Thank you