VayuBench and VayuChat: Executable Benchmarking and Deployment of LLMs for Multi-Dataset Air Quality Analytics

VEDANT ACHARYA*, Indian Institute of Technology Gandhinagar, India
ABHAY PISHARODI*, Indian Institute of Technology Gandhinagar, India
RATNESH PASI, Indian Institute of Information Technology Surat, India
RISHABH MONDAL, Indian Institute of Technology Gandhinagar, India
NIPUN BATRA, Indian Institute of Technology Gandhinagar, India

Air pollution causes over 1.6 million premature deaths annually in India. Yet, decision-makers face persistent barriers in turning diverse tabular data on air pollution, population, and funding into actionable insights. Existing tools demand technical expertise, offer shallow visualizations, or rely on static dashboards, leaving policy questions unresolved. Large language models (LLMs) offer a potential alternative by translating natural-language questions into structured, multi-dataset analyses; however, their reliability for such domain-specific tasks remains unknown. We present VayuBench, to our knowledge, the first executable benchmark for air-quality analytics. It comprises 5,000 natural-language queries paired with verified Python code across seven query categories: spatial, temporal, spatio-temporal, population-based, area-based, funding-related and specific pattern queries over multiple real-world datasets. We evaluate 13 open-source LLMs under a unified, schema-aware protocol. While Qwen3-Coder-30B attains the strongest performance, frequent column-name and variable errors highlight risks for smaller models. To bridge evaluation with practice, we deploy VayuChat, an interactive assistant that delivers real-time, code-backed analysis for Indian policymakers and citizens. Together, VayuBench and VayuChat demonstrate a reproducible pathway from benchmark to verified execution to deployment, establishing the foundations for trustworthy LLM-driven decision support in environmental monitoring.

CCS Concepts: • **Computing methodologies** → **Machine learning**; *Natural language processing*; • **Information systems** → *Data mining*.

Additional Key Words and Phrases: Air Quality, Large Language Models, Code Generation, Environmental Data, Benchmark, Python

ACM Reference Format:
Vedant Acharya, Abhay Pisharodi, Ratnesh Pasi, Rishabh Mondal, and Nipun Batra. 2025. VayuBench and VayuChat: Executable Benchmarking and Deployment of LLMs for Multi-Dataset Air Quality Analytics. In *13th ACM IKDD International Conference on Data Science (CODS 2025), December 17–20, 2025, Pune, India.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3799830.3799884

*Both authors contributed equally to this research.

Authors' Contact Information: Vedant Acharya, vedant.acharya@iitgn.ac.in, Indian Institute of Technology Gandhinagar, Gandhinagar, Gujarat, India; Abhay Pisharodi, abhay.pisharodi@iitgn.ac.in, Indian Institute of Technology Gandhinagar, Gandhinagar, Gujarat, India; Ratnesh Pasi, ratneshpasi03@gmail.com, Indian Institute of Information Technology Surat, Surat, Gujarat, India; Rishabh Mondal, rishabh.mondal@iitgn.ac.in, Indian Institute of Technology Gandhinagar, Gandhinagar, Gujarat, India; Nipun Batra, nipun.batra@iitgn.ac.in, Indian Institute of Technology Gandhinagar, Gandhinagar, Gujarat, India.

Manuscript submitted to ACM

1 Introduction

Air pollution poses a severe public health crisis in India, contributing to more than 1.6 million premature deaths annually [16]. Exposure to fine particulate matter ($PM_{2.5}$) reduces average life expectancy in India by over five years [31]. To address this crisis, CPCB[1] operates a network of over 500 monitoring stations [6], providing continuous measurements of $PM_{2.5}$ and other pollutants. Yet, monitoring is only the first step; turning raw readings into timely, actionable insights for policy remains an unsolved problem [11].

Even seemingly simple questions such as "How did $PM_{2.5}$ levels change in Delhi last year?" remain difficult for most stakeholders to answer directly. More complex, policy-relevant queries, for example, "Which cities reduced $PM_{2.5}$ the most relative to their funding from the NCAP[2] scheme?" or "Did Tier-1 cities improve faster than Tier-2 cities after 2020?", requires integrating pollution, funding, and demographic datasets. Available tools either demand significant technical expertise (e.g. R, SQL, Python) [5], offer only limited one-click visualizations without analytical depth [13], or focus narrowly on a subset of tasks (e.g., static dashboards for visualization). As a result, stakeholders often rely on data analysts to perform such analyses, slowing decision-making and restricting access to those with specialist support.

Large language models (LLMs) have emerged as powerful tools for translating natural-language requests into executable analyses [4, 30], offering the potential to make environmental data as accessible as a search query. In principle, an LLM could answer air quality-related questions like those mentioned above by identifying the right datasets, selecting appropriate statistical operations, and generating correct Python code to produce the answer. While prior work has evaluated LLMs for general code generation and data science tasks, their performance on domain-specific, multi-dataset environmental analytics, such as integrating pollution, funding, and demographic data for policy-relevant air quality queries, remains unexplored. Recent advances in instruction-following [30] and zero-shot reasoning over tabular data [39] suggest strong potential, but rigorous, domain-grounded evaluation is essential before such systems can be trusted in high-stakes decision support.

To address this gap, we introduce **VayuBench**, the first executable benchmark for environmental analytics, comprising 5,000 natural-language air quality queries paired with verified Python code across seven query categories: Area-Based Aggregation (AB), Funding Related Questions (FQ), Population-Based Exposure (PB), Spatial Aggregation (SA), Spatio-Temporal Aggregation (STA), Specific Patterns (SP), Temporal Trends (TT). These categories were derived from a comprehensive review of policy reports, academic studies, regulatory frameworks, public health assessments, think tank publications, environmental monitoring guidelines, stakeholder consultations with air quality experts, and interviews with environmental data practitioners, ensuring they reflect real-world decision-making needs. Built from Indian datasets: CPCB pollutant measurement, NCAP city-level funding records, and Indian state population area demographics, VayuBench evaluates whether LLMs can accurately integrate multiple tabular sources, select appropriate analytical operations, and produce correct, runnable code. Complementing the benchmark, we deploy **VayuChat**, an interactive assistant evaluated by the VayuBench framework, enabling policymakers, analysts, and citizens to query air-quality data in natural language and receive code-backed, reproducible answers in real time.

Evaluating 13 open-source LLMs on VayuBench reveals substantial performance variation: Qwen3-Coder-30B achieves a pass@1 value of 0.79, followed by Qwen3-32B with a pass@1 value of 0.78. Common syntactic error types

[1]CPCB is a statutory body under the Government of India that monitors and regulates water and air pollution. It has installed over 500 sensors nationwide under the National Air Monitoring Programme (NAMP).
[2]The National Clean Air Programme (NCAP), launched by the Government of India in 2019, has released over Rs. 9,650 crore to 131 non-attainment cities between FY 2019–20 and FY 2023–24 to improve urban air quality.

Table 1. Comparison of VayuBench with representative benchmarks and tools. VayuBench uniquely combines domain grounding in environmental data, executable Python code evaluation, multi-dataset integration, and spatio-temporal reasoning, with direct deployment through VayuChat.

Area	Benchmarks / Tools	Domain grounded	Executable Code	Multi-dataset	Spatio-temporal Reasoning	Deployed System
Text-to-SQL Generation	Spider [38], Text2SQL [21]	✗	✓(SQL)	✓	✗	✗
Medical Structured QA	MIMICSQL [24], EHRSQL [23]	✓(healthcare)	✓(SQL)	✓	✗	✗
General Code Generation	HumanEval [8], MBPP [3]	✗	✓	✗	✗	✗
Data Science Tasks	DS-1000 [22]	✗	✓(Python)	✗	✗	✗
Table-to-Text	TAPAS [18], ToTTo [32]	✗	✗	✓	✗	✗
Table QA	GRI-QA [12]	✓(environment)	✗	✓	✗	✗
Environmental QA / Tools	OpenAQ, Naqi [13], openair [5]	✓(environment)	✗	✗	✓	✗
Air-quality QA	**VayuBench + VayuChat**	✓(environment)	✓(Python)	✓	✓	✓

include incorrect column selection, name errors, and syntax violations. These errors directly undermine trust in smaller LLMs in policy contexts. In this paper, we contribute:

(1) **Executable benchmark:** VayuBench, the first domain-specific benchmark for air-quality analytics, pairing 5,000 natural language queries with verified Python code over multiple real-world datasets and systematically defined complex query categories.
(2) **Systematic LLM evaluation:** a unified schema-aware prompting protocol with machine-verifiable metrics (exec@1, pass@k), revealing significant capability gaps across models.
(3) **System deployment:** VayuChat, an interactive chatbot demonstrating how VayuBench can translate into accessible, trustworthy decision support for air quality policy and analysis.

VayuBench and **VayuChat** provide a reproducible pathway from benchmarking to deployment, enabling trustworthy LLM-based decision support in environmental monitoring. **Benchmark**: https://github.com/sustainability-lab/VayuBench, **Deployment**: https://huggingface.co/spaces/SustainabilityLabIITGN/VayuChat.

2 Related Work

We now discuss how VayuBench and VayuChat extend existing research in the fields of text-to-structured-query generation, code generation and data science benchmarks, table-to-text generation, and environmental data analysis and decision support tools.

Text-to-Structured-Query Generation. Mapping natural language to executable queries has been widely studied in Text2SQL and related settings. Text2SQL benchmarks such as WikiSQL [42], Spider [38] and its successors [21], as well as multi-turn datasets like CoSQL [41], enable complex SQL generation over multi-table or conversational settings. Domain-specific datasets such as MIMICSQL [24] and EHRSQL [23] target medical records. However, these resources do not cover environmental or policy-relevant datasets, lack spatio-temporal reasoning, and do not evaluate Python code generation.

Code Generation and Data Science Benchmarks. General-purpose code generation benchmarks such as HumanEval [8], MBPP [2], and CodeXGLUE [26] assess correctness on small, isolated code snippets. Larger-scale efforts such as CodeNet [35] and APPS [17] introduce more diverse and challenging programming problems, while competitive programming datasets like AlphaCode's CodeContests [25] push problem-solving complexity further. DS-1000 [22] extends evaluation to Python-based data science tasks, but lacks multi-table integration and spatio-temporal reasoning. Tool-augmented reasoning frameworks such as ReAct [40] and Toolformer [36] enable LLMs to interact with APIs and

libraries, yet are not designed for domain-grounded, multi-table analytics. In contrast, VayuBench targets executable, policy-relevant analytics grounded in real, multi-source environmental datasets.

Table-to-Text Generation. Early benchmarks such as ToTTo (Controlled Table-to-Text Generation) [32] and Logic-NLG (Logical Natural Language Generation) [10] focus on generating factually correct natural language descriptions from tables. Table-question-answering datasets like TAPAS (Table Parsing with BERT) [18] and WikiTableQuestions [34] extend this to reasoning over tabular inputs, while FeTaQA (Few-shot Table-to-Text QA) [28] and SciGen [27] emphasize explanatory or domain-specific table summarization. These resources advance factual and logical inference from structured data, yet they do not assess whether an LLM can produce correct, runnable analysis code, nor do they evaluate integration of multiple heterogeneous datasets for policy-relevant, multi-table queries as in *VayuBench*.

Environmental Data Analysis. Air quality research has traditionally emphasized predictive modelling and sensor calibration rather than natural-language-to-analysis interfaces. Physics-guided approaches such as AirPhysNet [19] improve forecasting accuracy by embedding atmospheric priors, while curated datasets like PurpleAirSF [43] enable reproducible prediction studies. Sensor correction techniques, including VELI [14] and low-cost calibration methods [37], enhance data quality for downstream analysis. While these contributions advance environmental monitoring and modelling, they do not address accessibility for non-technical stakeholders or evaluate LLMs on generating correct, executable multi-dataset analytics, as supported in *VayuBench*.

Environmental Decision-Support Tools. Decision-support platforms such as OpenAQ [29], Naqi [13], and the `openair` R package [5] provide valuable access to air quality data, visualisation, and reporting tools. However, these systems often require substantial technical expertise, limit users to a predefined set of analyses, or remain restricted to static dashboards. None provide a systematic, executable benchmark for assessing automated reasoning capabilities, nor an integrated, interactive deployment such as *VayuChat*.

VayuBench and VayuChat in Context. As summarised in Table 1, VayuBench uniquely combines real multi-source environmental datasets, seven systematically defined query categories, executable Python code evaluation, and a deployed assistant (VayuChat). This addresses gaps in domain grounding, execution, and policy relevance not covered by existing benchmarks. VayuChat, hosted on Hugging Face Spaces[3], provides a natural-language interface to query Indian air quality data, executing LLM-generated Python code and returning results through visualizations and explanations.

3 VayuBench: An Executable Benchmark for Environmental Analytics

In this section, we first discuss the design goals of VayuBench (Section 3.1). We then describe the dataset information and schema (Sections 3.2 and 3.3). Based on these, we present the air-quality query categories (Section 3.4). Finally, we elaborate on the construction of the benchmark with high-quality query–code pairs (Section 3.5) and the evaluation protocol (Section 3.7) used for evaluating the LLMs.

3.1 Stakeholder Discussion, Design Goals, and Survey

Preliminary Expert Discussions. We engaged air-quality scientists, academic researchers, and policy think tanks in semi-structured interviews and informal surveys. Three priorities emerged as essential for any decision-support system: (i) *transparency of analytical steps*, (ii) *reproducibility of outputs*, and (iii) *accessibility for non-technical stakeholders*.

[3]https://huggingface.co/spaces/SustainabilityLabIITGN/VayuChat

These were not abstract preferences — each was linked to specific pain points, such as the inability to verify analyses in current dashboards, or the difficulty of joining multiple datasets without technical skills.

Benchmark Design Objectives. We translated these priorities into three benchmark design goals:

(1) **Stakeholder Coverage (G1):** Include query types that reflect the genuine decision-making needs of policymakers, scientists, and citizens.
(2) **Executable Reliability (G2):** Ensure that each query can test whether LLMs produce not just syntactically correct but functionally valid and verifiable Python code, supporting transparency and reproducibility.
(3) **Extensible Accessibility (G3):** Make the framework flexible enough to incorporate new datasets and question types, enabling future expansions and deployed systems for non-technical users.

Initial Question Set and Survey Validation. Guided by these goals, we drafted 65 seed questions across seven categories, all rooted in realistic policy and research scenarios. Example queries included:

- *Spatial:* Which state had the highest average $PM_{2.5}$ concentration?
- *Temporal:* Which year recorded the highest annual average $PM_{2.5}$?
- *Funding-related:* Which cities improved most relative to their NCAP funding?
- *Population-based:* What percentage of the population was exposed to $PM_{2.5}$ above 60 $\mu g/m^3$?

A structured survey of policymakers, analysts, and practitioners rated the importance of each category (1–5 scale), suggested missing query types, and provided real-world examples. This process directly informed both the content of the benchmark and the datasets it required.

3.2 Benchmark Datasets

From Priorities to Data. Survey feedback and discussions converged on three essential dimensions of air-quality decision-making: (i) accurate spatio-temporal pollutant measurements, (ii) linkage of interventions (funding) to outcomes, and (iii) population and area context for equity analysis. We curated three datasets to satisfy these requirements and align with G1–G3.

Air Quality Dataset (CPCB). Daily station-level $PM_{2.5}$ and PM_{10} measurements (2017–2024) from the Central Pollution Control Board address G1 by enabling detailed spatial, temporal, and spatio-temporal queries. Stakeholders identified this as the authoritative baseline for transparency.

Funding Dataset (NCAP). City- and state-level allocations and utilization under the National Clean Air Programme (2019–2022) meet G2 by enabling reproducible analysis of the relationship between interventions and outcomes.

State Demographics Dataset. Population, area, and union territory status for all states support G3 by enabling equity-focused metrics such as population-weighted exposure and area-normalized trends, making results more interpretable to non-technical audiences.

Integration and Processing. All three datasets were cleaned, standardized into Pandas DataFrames, and serialized as `.pkl` files for efficient, reproducible execution within the benchmark. This integration ensures that every query in VayuBench is answerable with transparent, reproducible, and stakeholder-relevant data.

3.3 Data Schema Definition

Motivation. To operationalize stakeholder needs and benchmark design goals (Section 3.1), each dataset was cleaned, normalized, and mapped into a well-defined schema. Standardizing the columns ensures transparency in how queries

are executed (supports reproducibility), enables LLMs to reason over consistent field names (supports evaluation), and allows direct integration across multiple datasets (supports accessibility and extensibility).

Air Quality Dataset (CPCB). Table 2 details the schema of daily station-level pollutant data. Columns capture temporal information (timestamps), spatial attributes (city, state, coordinates), and pollutant values ($PM_{2.5}$, PM_{10}). This design supports spatio-temporal queries such as: *"Which state had the highest average $PM_{2.5}$ in 2022?"* It directly addresses stakeholder calls for transparent and reproducible pollutant trend analysis (G1, G2).

Table 2. Schema of the Air Quality Dataset, supporting spatio-temporal analysis of pollutant trends across India.

Column	Data Type	Description
Timestamp	datetime64[ns]	Measurement date
station	object	Monitoring station name
PM2.5	float64	Daily mean PM2.5 ($\mu g/m^3$)
PM10	float64	Daily mean PM10 ($\mu g/m^3$)
address	object	Station address
city	object	City of station
latitude	float64	Station latitude
longitude	float64	Station longitude
state	object	State or union territory

Funding Dataset (NCAP). Table 3 defines the schema of NCAP funding allocations and utilization. Stakeholders repeatedly requested transparency in linking resource flows to outcomes. This dataset enables reproducible funding–outcome queries, such as: *"Which cities improved most relative to their funding?"*, directly addressing reproducibility and accountability concerns (G2).

Table 3. Schema of the NCAP Funding Dataset, enabling transparency in financial allocations and utilization.

Column	Data Type	Description
S. No.	int64	Serial number
state	object	State or UT of city
city	object	NCAP-funded city
Amount released		**Funding (in crores)**
FY 2019–20	float64	In 2019–2020
FY 2020–21	float64	In 2020–2021
FY 2021–22	float64	In 2021–2022
Total fund released	float64	Total allocated funds
Utilisation as on June 2022	float64	Funds used as of June 2022

State Demographics Dataset. Table 4 specifies the schema of demographic attributes (population, area, union territory status). This dataset supports queries on equity and accessibility, e.g., *"What proportion of the population was exposed to $PM_{2.5}$ above 60 $\mu g/m^3$?"*. It operationalizes the stakeholder priority of making results interpretable in terms of population exposure and equity (G3).

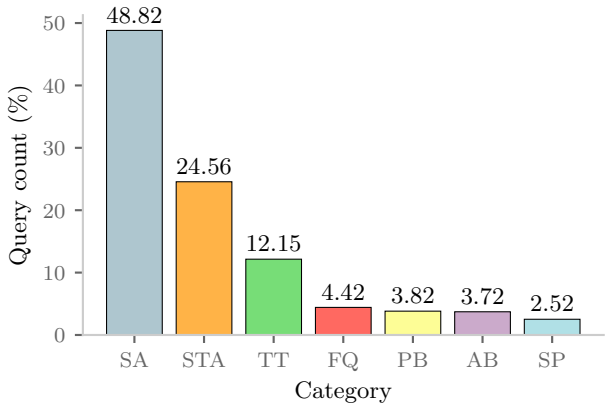

Fig. 1. Distribution of the 5,000 benchmark queries across categories. Spatial, spatio-temporal, and temporal queries dominate due to richer variability in station- and city-level data, while funding- and population-linked queries are fewer, reflecting their narrower schema coverage.

Table 4. Schema of the State Demographics Dataset, enabling population- and area-normalized analyses.

Column	Data Type	Description
state	object	Indian state or union territory
population	int64	Population
area (km^2)	int64	Geographical area (km^2)
isUnionTerritory	bool	Whether the region is a UT

3.4 Benchmark Query Types

Building on stakeholder discussions (Section 3.1), we distilled seven query categories that reflect the most common and consequential analytical tasks for air-quality decision-making. Proportion of each category is shown in Figure 1. Each category encodes a distinct reasoning skill required of LLMs, grounded in real policy and research use cases.

(1) **Area-based Aggregation (AB):** Queries normalizing by geographical area, reflecting fairness in monitoring coverage. *Example: Which state has the fewest stations per km^2 ?*
(2) **Funding-related Questions (FQ):** Queries on NCAP allocations and utilization, critical for financial accountability. *Example: Which year saw the highest average fund release across cities?*
(3) **Population-based Exposure (PB):** Queries joining air quality with demographics to assess exposure burden. *Example: What fraction of the population lives in regions exceeding WHO $PM_{2.5}$ limits?*
(4) **Spatial Aggregation (SA):** Location-based summaries at fixed times. *Example: Which Delhi station had the highest mean $PM_{2.5}$ in Jan 2022?*
(5) **Spatio-Temporal Aggregation (STA):** Summaries across both space and time. *Example: Which state recorded the worst PM_{10} levels in summer 2022?*
(6) **Specific Patterns (SP):** Detection of rule-based violations over short windows. *Example: Over the past five years, how many days did Mumbai exceed WHO $PM_{2.5}$ limits?*
(7) **Temporal Trends (TT):** Longitudinal patterns in pollution. *Example: How did $PM_{2.5}$ evolve in Lucknow across 2021?*

3.5 Benchmark Construction

To ensure VayuBench is both realistic and reproducible, we adopted a four-step pipeline (Figure 2) that systematically expands a small seed set of real-world questions into a large, diverse benchmark:

(1) **Seed Question Collection.** We began with 65 base questions grounded in stakeholder needs across the seven categories (Section 3.4). Each question was mapped to the required data types and validated against available columns. This ensured the benchmark remained tied to authentic use cases rather than synthetic queries.
(2) **Template Design.** From these seeds, we abstracted hand-crafted templates with placeholders for valid method–column pairs. Templates directly mirror typical Pandas operations (`groupby`, `mean`, `max`, etc.), enabling transparent mapping from natural language to code. This step guaranteed systematic coverage of reasoning patterns rather than ad-hoc phrasing.
(3) **Systematic Expansion and Sampling.** Instantiating all valid method–column combinations produced ∼26K candidate queries. From these, we sampled 5K queries to maximize diversity across locations, metrics, and temporal spans, while preventing redundancy. This balanced scale with usability.
(4) **Paraphrasing for Naturalness.** To reduce annotation artifacts and improve linguistic variety, each sampled query was paraphrased using Gemini and then manually verified. We retained only paraphrases that preserved parameters and semantics. This step prevents models from overfitting to rigid templates and better reflects real-world phrasing.

3.6 System Prompt Schema

All prompts follow a standardized schema to ensure consistency and reproducibility. The model is explicitly instructed to act as an *air quality expert Python code generator*. Inputs are restricted to three DataFrames: `data` (daily air quality, 2017–2024, with station, city, state, and pollutant columns), `states_data` (state-level demographics and area), and `ncap_funding_data` (city-level NCAP funding, 2019–2022). The schema enforces: (i) A fixed function signature: `get_response(data, states_data, ncap_funding_data)`. (ii) Output restricted to a single scalar value (no DataFrames, tuples, or plots). (iii) Explicit imports and code-only responses, wrapped in `<code>...</code>`. (iv) Uniform evaluation using `exec@1` and `pass@k`, applied directly to the generated function. This schema eliminates prompt ambiguity, guarantees comparability across models, and provides a reproducible interface for execution. Extended prompt templates and full examples are available in our GitHub repository.

3.7 Evaluation Protocol

Sampling strategy. For each query, we sample $n = 5$ completions using temperature sampling (temperature = 0.8, top-p = 0.95). Prior work (e.g., [9]) often uses $n = 20$–100, but this is computationally prohibitive for our large dataset and multi-model comparison. Trial runs showed that after n = 5, the improvement became very small, so choosing (n = 5) gives a good balance between variety and efficiency.
We used the following evaluation metrics:

- **pass@k:** Fraction of queries where at least one of the top-k samples generated by the model passes all tests. For query i with c_i correct out of n, pass@k = $\frac{1}{N}\sum_{i=1}^{N}\left(1-\frac{\binom{n-c_i}{k}}{\binom{n}{k}}\right)$.
- **exec@1:** Measures the proportion of samples that execute without error, regardless of correctness. For query i with c_i successful executions out of n, exec@1 = $\frac{1}{N}\sum_{i=1}^{N}\frac{c_i}{n}$.

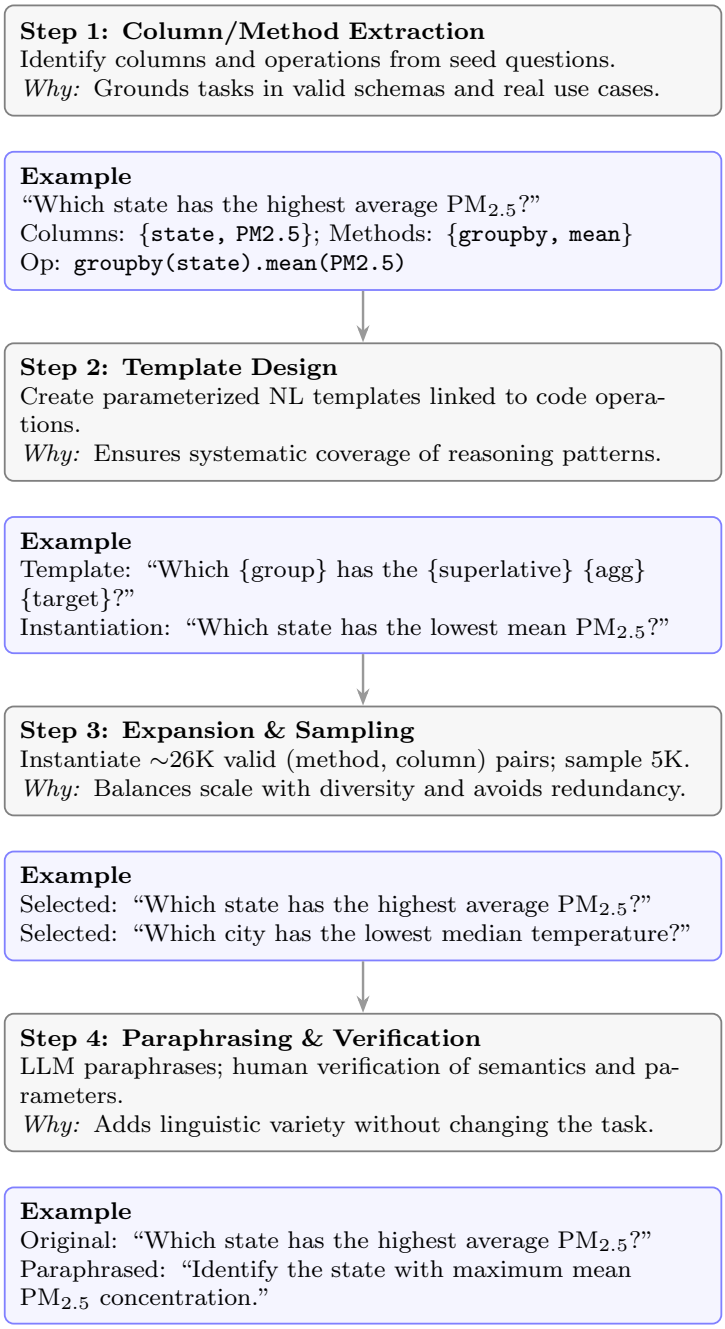

Fig. 2. Four-step construction pipeline for VayuBench: (1) grounding in schemas; (2) systematic template design; (3) controlled yet diverse scaling; (4) natural phrasing with human verification.

Table 5. Overall evaluation results of LLMs on VayuBench across 5,000 questions. The table reports execution and passing accuracy (exec@1, pass@1, pass@2), error type distributions and overall error rate. Syntax, Column, Name, and Other denote the proportion of generated code samples exhibiting each error type. The 1st, 2nd, and 3rd place values across columns are highlighted in red, green, and blue, respectively. (Ties in error metrics are broken using the exec@1 column)

Model	**Params (B)**	Accuracy Metrics (↑)			Error Metrics (↓)				
		exec@1	pass@1	pass@2	Syntax	Column	Name	Other	ErrorRate
Llama3.2	1.0	0.04	0.00	0.01	0.58	0.25	0.12	0.02	0.97
Qwen2.5-Coder	1.5	0.47	0.08	0.12	0.02	0.49	0.01	0.01	0.53
Qwen3	1.7	0.46	0.05	0.08	0.02	0.49	0.01	0.01	0.54
Llama3.2	3.0	0.17	0.04	0.07	0.00	0.20	0.62	0.01	0.83
Qwen2.5-Coder	3.0	0.73	0.33	0.47	0.00	0.25	0.00	0.01	0.26
Deepseek-Coder	6.7	0.77	0.48	0.54	0.00	0.19	0.00	0.03	0.23
Mistral	7.0	0.24	0.03	0.05	0.17	0.56	0.00	0.03	0.76
Qwen2.5-Coder	7.0	0.79	0.41	0.54	0.00	0.20	0.00	0.00	0.20
Codellama	13.0	0.64	0.25	0.32	0.00	0.35	0.00	0.00	0.36
Qwen2.5-Coder	14.0	0.90	0.69	0.74	0.00	0.05	0.00	0.01	0.06
GPT-OSS	20.0	0.88	0.56	0.69	0.03	0.06	0.03	0.00	0.12
Qwen3-Coder	30.0	0.99	0.79	0.81	0.00	0.01	0.00	0.00	0.01
Qwen3	32.0	0.98	0.78	0.81	0.00	0.01	0.00	0.00	0.01

Table 6. Category-wise evaluation of models on VayuBench. Columns show E1 (*exec@1*), P1 (*pass@1*), and P2 (*pass@2*) for each category, with category abbreviations defined in Section 3.4. The 1st, 2nd, and 3rd values in each column are highlighted in red, green, and blue, respectively.

Model	**Params (B)**	AB (↑)			FQ (↑)			PB (↑)			SA (↑)			STA (↑)			SP (↑)			TT (↑)		
		E1	P1	P2	E1	P1	P2	E1	P1	P2	E1	P1	P2	E1	P1	P2	E1	P1	P2	E1	P1	P2
Llama3.2	1.0	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.05	0.01	0.01	0.04	0.00	0.00	0.02	0.00	0.00	0.02	0.00	0.00
Qwen2.5-Coder	1.5	0.18	0.03	0.05	0.22	0.01	0.02	0.26	0.01	0.02	0.56	0.11	0.17	0.42	0.06	0.09	0.43	0.09	0.15	0.48	0.08	0.12
Qwen3	1.7	0.38	0.05	0.08	0.34	0.02	0.04	0.44	0.02	0.03	0.34	0.05	0.08	0.64	0.05	0.08	0.44	0.07	0.12	0.69	0.10	0.12
Llama3.2	3.0	0.06	0.01	0.03	0.14	0.01	0.02	0.09	0.01	0.02	0.19	0.06	0.11	0.18	0.01	0.03	0.10	0.01	0.01	0.13	0.02	0.04
Qwen2.5-Coder	3.0	0.38	0.15	0.24	0.30	0.04	0.06	0.43	0.07	0.12	0.78	0.52	0.72	0.81	0.15	0.22	0.87	0.33	0.42	0.68	0.19	0.28
Deepseek-Coder	6.7	0.86	0.36	0.46	0.83	0.12	0.17	0.83	0.18	0.24	0.78	0.70	0.76	0.72	0.21	0.25	0.92	0.43	0.48	0.73	0.41	0.46
Mistral	7.0	0.19	0.00	0.01	0.30	0.02	0.03	0.19	0.01	0.02	0.17	0.02	0.03	0.23	0.03	0.05	0.47	0.10	0.16	0.47	0.10	0.12
Qwen2.5-Coder	7.0	0.70	0.38	0.56	0.72	0.13	0.20	0.61	0.18	0.29	0.89	0.62	0.77	0.79	0.19	0.27	0.76	0.35	0.48	0.47	0.24	0.36
Codellama	13.0	0.48	0.15	0.24	0.63	0.08	0.13	0.63	0.08	0.12	0.78	0.39	0.51	0.41	0.05	0.07	0.72	0.21	0.29	0.62	0.20	0.24
Qwen2.5-Coder	14.0	0.97	0.85	0.92	0.96	0.32	0.43	0.89	0.64	0.75	0.98	0.94	0.96	0.90	0.34	0.41	0.98	0.63	0.70	0.86	0.47	0.58
GPT-OSS	20.0	0.77	0.57	0.80	0.90	0.34	0.48	0.81	0.53	0.72	0.89	0.78	0.93	0.88	0.26	0.37	0.81	0.06	0.08	0.91	0.44	0.56
Qwen-3-Coder	30.0	1.00	0.87	0.90	1.00	0.46	0.53	0.95	0.80	0.85	1.00	0.98	0.98	1.00	0.50	0.53	0.81	0.56	0.60	1.00	0.76	0.78
Qwen3	32.0	0.96	0.85	0.90	0.96	0.51	0.61	0.94	0.73	0.82	0.99	0.96	0.98	0.99	0.54	0.58	0.98	0.53	0.59	0.99	0.66	0.68

4 Results and Analysis

4.1 Models Evaluated

To probe both specialization and scaling, we evaluated a diverse set of open-source models on HuggingFace (API models excluded due to infeasible sampling costs). We used temperature=0.7, topP=0.8, topK=20, and minP=0 for all our experiments.

- **Code-centric vs. general-purpose:** Code-specialized models (Qwen2.5-Coder, Qwen3-Coder) are contrasted with general LLMs (Mistral, Llama-3.2, GPT-OSS-20B) to measure the benefit of fine-tuning on code generation.
- **Small vs. large:** Within families, we include lighter models (e.g., Qwen2.5-Coder-1.5B) and their larger counterparts (Qwen3-Coder-30B) to isolate scaling effects.

This design allows us to ask: *Does scale or specialization matter more for reliable execution on structured AQ datasets?*

4.2 Performance Trends

Table 5 shows aggregate metrics, with category-wise results in Table 6. Key findings:

(1) **Scaling improves reliability.** Larger Qwen models dominate, with Qwen3-Coder-30B achieving the best overall exec@1 (0.99) and pass@1 (0.79).
(2) **Specialization matters more than size in schema-heavy tasks.** Qwen2.5-Coder-14B outperforms larger generalists in categories such as AB and SP.
(3) **General-purpose models lag.** Mistral and Llama-3.2 consistently underperform, highlighting limited zero-shot transfer without code tuning.
(4) **Model biases differ.** Qwen-2.5-Coder-14B excels in specific patterns (SP) despite weaker overall pass@1, indicating distinct inductive strengths.

4.3 Impact of Model Size and Architecture

As model size increases, both **exec@1** and **pass@1** performance generally improve.
Smaller models exhibit low exec@1 (< 0.5) and pass@1 (< 0.5). Qwen2.5-3B and Qwen3-1.7B outperform comparable Llama3.2-3B and Llama3.2-1B models, likely because Qwen is more extensively fine-tuned on code generation tasks.

Mid-sized models show a significant gain in exec@1 GPT-OSS-20B (0.88) and Qwen-2.5-Coder-14B (0.90) over smaller models, reflecting improved syntax awareness.

Very large models like Qwen3-Coder-30B, achieve the highest exec@1 (0.99) and pass@1 (0.79), indicating an approximately linear relationship between performance and model parameters.

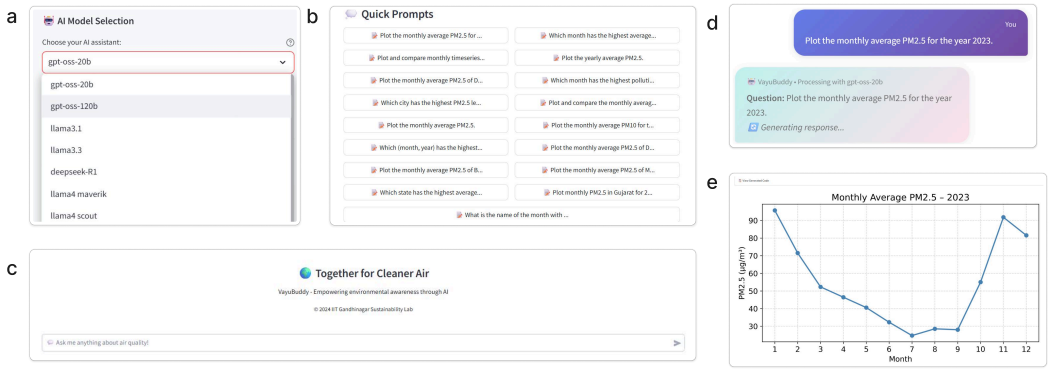

Fig. 3. VayuChat interface for air quality analysis. It supports: (a) model selection (e.g., GPT-OSS, open-source LLMs), (b) predefined query prompts, (c) custom natural language input, (d) code generation and execution, and (e) outputs with visualizations and statistics.

4.4 Error Definitions

We quantify errors using the **error rate**: for N queries, each with n samples, if e_i samples of query i fail syntactically:
Error rate $= \frac{1}{N} \sum_{i=1}^{N} \frac{e_i}{n}$

Syntactic errors: Code cannot run due to structural issues.

- *Syntax:* Invalid grammar (e.g., unmatched brackets).
- *Column:* Non-existent/misspelt column names (e.g., `PM25` vs `PM2.5`).
- *Name:* Undefined variables/objects (e.g., calling `pd.merge()` without importing `pandas`).
- *Other:* Miscellaneous (e.g., division by zero).

Semantic errors: Code executes but logic is incorrect (e.g., yearly instead of monthly aggregation). Examples are mentioned in the Table 7 and 8.

4.5 Error Analysis

Table 5 decomposes failures by error type. **Column errors dominate** across models, accounting for nearly half of all failures. This indicates that schema alignment (e.g., matching variable or field names) remains the primary bottleneck. **Syntax errors** are rare overall, but seen in smaller models (e.g., LLaMa3.2-1B, error rate 0.58), highlighting weak Python fluency at low parameter counts. **Name errors** concentrate in LLaMa3.2-3B (0.62), reflecting unstable variable handling, while **Other errors** (e.g., division by zero) remain marginal. Comparing models, LLaMa3.2-1B records the highest overall error rate (0.97), followed by LLaMa3.2-3B (0.83) and Mistral-7B (0.76). In contrast, large-scale models (Qwen2.5-14B, GPT-OSS-20B, Qwen3-32B) reduce all error categories, achieving error rates ≤ 0.12. Interestingly, the specialized *Qwen3-Coder-30B* attains top accuracy (exec@1 = 0.99) but does not eliminate column errors, suggesting that schema reasoning, rather than syntax, is the limiting factor.

5 VayuChat: Deployed Application

As mentioned in Figure 3, VayuChat demonstrates the practical utility of VayuBench as an LLM-powered conversational agent that bridges raw air quality datasets with actionable policy insights. It enables policymakers, analysts, and citizens to query air quality data in natural language, while each response is backed by executable Python code for transparency and reproducibility. User feedback highlights two key strengths: (i) accessibility of complex data for non-technical users and (ii) code outputs that allow experts to verify and refine analyses. In this section, we first describe VayuChat's system architecture and its use cases.

5.1 System Architecture

VayuChat is deployed as a web application on Hugging Face Spaces[4], offering an intuitive interface for interactive querying of Indian air quality data (Figure 3).

User Interface. A lightweight, browser-based interface supports free-text queries, guided dataset exploration, and interactive visualization outputs. Pre-loaded example queries orient new users, while advanced users can submit arbitrary questions. Results are presented in text, tables, and plots.

LLM Orchestration. Each query is embedded into a structured prompt containing: (i) the user query, (ii) dataset schema descriptions, and (iii) explicit constraints (single scalar or visualization outputs, strict function signature, explicit imports). This structure enforces safe code generation. The system routes requests via Groq and Gemini APIs, enabling fast open-source LLMs (benchmarked in VayuBench) alongside advanced commercial models. A model-switching interface allows side-by-side comparison across LLMs for accuracy, robustness, and latency.

[4]https://huggingface.co/spaces/SustainabilityLabIITGN/VayuChat

Computation and Verification. Generated code executes in a secure, containerized Python sandbox preloaded with datasets and core libraries (Pandas, NumPy, Matplotlib). Resource limits and isolation ensure safe execution. Failures trigger automated repair attempts or explicit error messages. Outputs: statistical summaries, plots, or text. A verification layer checks code against schema validity and benchmark-derived constraints (Section 3.5), reducing hallucinations and enforcing reproducibility.

5.2 Representative Analytical Workflows

VayuChat supports diverse analytical workflows without requiring coding expertise. It handles temporal analysis (e.g., trends of $PM_{2.5}$ in Delhi over the past year), comparative analysis (e.g., monthly $PM_{2.5}$ trends for Mumbai vs. Bengaluru), spatial ranking (e.g., identifying the city with the highest average $PM_{2.5}$ in 2022), cross-domain integration (e.g., linking NCAP funding with pollution reduction), and demographic coupling (e.g., comparing air quality across high- and low-population states). Each response includes generated Python code, allowing experts to verify or extend the analysis while keeping results accessible to non-technical stakeholders.

Table 7. Aggregation error by LLM: grouping by `city` instead of `state`. Correct answer = 42241.

Q (AB): UT area with 3rd max PM2.5+PM10.
LLM (Incorrect):

```
city_pollution = data.groupby('city')[['PM2.5','PM10']].sum().reset_index()
```

Ground Truth (Correct):

```
state_averages = main_data.groupby('state')[['PM2.5','PM10']].mean()
```

Table 8. Spatial aggregation error: the model computed the 3rd-lowest percentile value instead of retrieving the associated state.

Q (SA): On Mar 31, 2023, which state had the 3rd-lowest 25th percentile for PM_{10}?
LLM Output: 31.3582
Correct Answer: Haryana
LLM (Incorrect):

```
third_lowest_percentile = state_percentiles.nsmallest(3).iloc[2]
```

Ground Truth (Correct):

```
print(data.iloc[2]["state"])
```

6 Limitations and Future Work

VayuBench and VayuChat currently rely on CPCB data; integrating OpenAQ, WHO's Global AQ Database, and meteorological inputs would enable richer causal analysis. Extending to multimodal sources (e.g., Sentinel-5P imagery, NCAP policy docs, GBD health data [20]) can broaden coverage. Methodologically, domain-adaptive finetuning [15], retrieval-augmented generation with curated corpora [7], and self-consistency prompting can improve factuality. Stronger NLU, runtime checks, and tool-based verification (e.g., Pandas Profiling, Great Expectations) are needed to reduce reasoning errors. New metrics and user feedback can better capture fairness and policy impact. Scaling to distributed backends (Ray [1], Particle [33]) and open-weight LLMs (LLaMA-3, Mistral) will enhance robustness and efficiency for deployment.

7 Conclusion

VayuBench and **VayuChat** together provide the first domain-grounded, executable benchmark and deployed interface for multi-dataset air quality analytics. Our contributions span reproducible evaluation, real-world deployment, and a demonstration of how LLMs can serve both the Machine Learning and Air Quality communities. This work provides a 5,000-query benchmark spanning seven categories, with templated and paraphrased questions grounded in real air-quality data, plus a reproducible evaluation framework with code annotations and pass@k metrics for rigorous LLM assessment in environmental analytics. For AQ, it links pollution data, geographic coverage, and funding records to show how ML pipelines can generate actionable insights, identify gaps, and inform evidence-based interventions. As a scalable policy analysis tool, the dataset and question-generation framework support environmental scientists, urban planners, and public health researchers in conducting interpretable, data-driven assessments and can inspire similar initiatives in other domains where environmental data intersects with public policy.

References

[1] Anyscale (Ray project). 2025. Ray: Scale Machine Learning & AI Computing. https://www.ray.io/. Accessed: 2025-08-18.

[2] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program Synthesis with Large Language Models. In *ICLR*.

[3] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732* (2021).

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL] https://arxiv.org/abs/2005.14165

[5] David C. Carslaw and Karl Ropkins. 2012. openair — An R package for air quality data analysis. *Environmental Modelling & Software* 27–28, 0 (2012), 52–61. doi:10.1016/j.envsoft.2011.09.008

[6] Central Pollution Control Board. 2025. CPCB - Overview and Functions. https://cpcb.nic.in/.

[7] Centre for Science and Environment. 2025. Air Quality and Public Health. https://www.cseindia.org/page/air-quality-and-public-health. Accessed: 2025-08-18.

[8] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG] https://arxiv.org/abs/2107.03374

[9] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG] https://arxiv.org/abs/2107.03374

[10] Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020. Logical Natural Language Generation from Open-Domain Tables. arXiv:2004.10404 [cs.CL] https://arxiv.org/abs/2004.10404

[11] Clarity Movement. 2022. How Open Access Air Pollution Data is Paving the Way for Greater Air Quality Awareness. https://www.clarity.io/blog/how-open-access-air-pollution-data-is-paving-the-way-for-greater-air-quality-awareness Accessed: 2025-05-15.

[12] Michele Luca Contalbo, Sara Pederzoli, Francesco Del Buono, Venturelli Valeria, Francesco Guerra, and Matteo Paganelli. 2025. GRI-QA: a Comprehensive Benchmark for Table Question Answering over Environmental Data. In *Findings of the Association for Computational Linguistics: ACL 2025*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, Vienna, Austria, 15764–15779. doi:10.18653/v1/2025.findings-acl.814

[13] CPCB. 2025. National air quality index Real-time data. https://airquality.cpcb.gov.in/AQI_India/ Accessed: 2025-05-15.

[14] Yahia Dalbah, Marcel Worring, and Yen-Chia Hsu. 2025. Veli: Unsupervised Method and Unified Benchmark for Low-Cost Air Quality Sensor Correction. arXiv:2508.02724 [eess.SP] https://arxiv.org/abs/2508.02724

[15] Aleksandra Eliseeva, Alexander Kovrigin, Ilia Kholkin, Egor Bogomolov, and Yaroslav Zharov. 2025. EnvBench: A Benchmark for Automated Environment Setup. arXiv:2503.14443 [cs.LG] https://arxiv.org/abs/2503.14443

[16] Energy Policy Institute at the University of Chicago (EPIC). 2023. AQLI India Fact Sheet. EPIC Report, August 2023, https://aqli.epic.uchicago.edu/wp-content/uploads/2023/08/India-FactSheet-2023_Final.pdf. Accessed: 2025-08-13.

[17] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Coding Challenge Competence with APPS. *arXiv preprint arXiv:2105.09938* (2021).

[18] Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349* (2020).

[19] Kethmi Hirushini Hettige, Jiahao Ji, Shili Xiang, Cheng Long, Gao Cong, and Jingyuan Wang. 2024. AirPhyNet: Harnessing Physics-Guided Neural Networks for Air Quality Prediction. arXiv:2402.03784 [cs.LG] https://arxiv.org/abs/2402.03784 Accepted by ICLR 2024.

[20] Institute for Health Metrics and Evaluation (IHME). 2025. Global Burden of Disease (GBD). https://www.healthdata.org/research-analysis/gbd. Accessed: 2025-08-18.

[21] Jinxiang Lai, Jie Zhang, Jun Liu, Jian Li, Xiaocheng Lu, and Song Guo. 2025. Spider: Any-to-Many Multimodal LLM. arXiv:2411.09439 [cs.CV] https://arxiv.org/abs/2411.09439

[22] Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2022. DS-1000: A Natural and Reliable Benchmark for Data Science Code Generation. arXiv:2211.11501 [cs.SE] https://arxiv.org/abs/2211.11501

[23] Gyubok Lee, Hyeonji Hwang, Seongsu Bae, Yeonsu Kwon, Woncheol Shin, Seongjun Yang, Minjoon Seo, Jong-Yeup Kim, and Edward Choi. 2023. EHRSQL: A Practical Text-to-SQL Benchmark for Electronic Health Records. arXiv:2301.07695 [cs.CL] https://arxiv.org/abs/2301.07695

[24] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs. arXiv:2305.03111 [cs.CL] https://arxiv.org/abs/2305.03111

[25] Yujia Li, David Choi, Matthew Gerstenberg, Usman Humayoun, Adria Recasens Jose, Pushmeet Kohli, June Koo, Young Lee, Lin Li, Wei Li, et al. 2022. Competition-Level Code Generation with AlphaCode. *Science* 378, 6624 (2022), 1092–1097.

[26] Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, Ming Gong, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, and Shujie Liu. 2021. CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation. arXiv:2102.04664 [cs.SE] https://arxiv.org/abs/2102.04664

[27] Nafise Sadat Moosavi, Lisa Fichtel, Sebastian Pado, and Iryna Gurevych. 2021. SciGen: A Dataset for Scientific Table-to-Text Generation. In *ACL*. 4376–4389.

[28] Linyong Nan, Jiacheng Yu, Xi Zhang, and Dragomir Radev. 2022. FeTaQA: Free-form Table Question Answering. In *ACL*. 1020–1034.

[29] OpenAQ Project. 2024. OpenAQ: Open Air Quality Data Platform. https://openaq.org/.

[30] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL] https://arxiv.org/abs/2203.02155

[31] Anamika Pandey, Michael Brauer, Maureen L Cropper, Kalpana Balakrishnan, Prashant Mathur, Sagnik Dey, Burak Turkgulu, G Anil Kumar, Mukesh Khare, Gufran Beig, et al. 2021. Health and economic impact of air pollution in the states of India: the Global Burden of Disease Study 2019. *The Lancet Planetary Health* 5, 1 (2021), e25–e38.

[32] Ankur P. Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A Controlled Table-To-Text Generation Dataset. arXiv:2004.14373 [cs.CL] https://arxiv.org/abs/2004.14373

[33] Particle. 2025. Particle: An Integrated IoT Platform-as-a-Service. https://www.particle.io/. Accessed: 2025-08-18.

[34] Panupong Pasupat and Percy Liang. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *ACL*. 1470–1480.

[35] Ruchir Puri, David Kung, et al. 2021. CodeNet: A Large-Scale AI for Code Dataset for Learning a Diversity of Coding Tasks. *arXiv preprint arXiv:2105.12655* (2021).

[36] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Shruti Bhosale, Xian Li Wang, Leon Derczynski, Mike Chrzanowski, Thomas Scialom, et al. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. *arXiv preprint arXiv:2302.04761* (2023).

[37] S. De Vito, E. Esposito, M. Salvato, O. Popoola, F. Formisano, R. Jones, and G. Di Francia. 2018. Calibrating chemical multisensory devices for real world applications: An in-depth comparison of quantitative Machine Learning approaches. *Sensors and Actuators B: Chemical* 255 (2018), 1191–1210. doi:10.1016/j.snb.2017.07.155

[38] Ping Wang, Tian Shi, and Chandan K. Reddy. 2020. Text-to-SQL Generation for Question Answering on Electronic Medical Records. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, ACL, Online.

[39] Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, Yong Jiang, and Wenjuan Han. 2023. Zero-Shot Information Extraction via Chatting with ChatGPT. arXiv:2302.10205 [cs.CL] https://arxiv.org/abs/2302.10205

[40] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629 [cs.CL] https://arxiv.org/abs/2210.03629

[41] Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Xingran Chen, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, and Dragomir Radev. 2019. SParC: Cross-Domain Semantic Parsing in Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4511–4523. doi:10.18653/v1/P19-1443

[42] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 231–240. doi:10.18653/v1/P17-1021

[43] Jingwei Zuo, Wenbin Li, Michele Baldo, and Hakim Hacid. 2023. Unleashing Realistic Air Quality Forecasting: Introducing the Ready-to-Use PurpleAirSF Dataset. arXiv:2306.13948 [cs.LG] https://arxiv.org/abs/2306.13948 Accepted by ACM SIGSPATIAL 2023.