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

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
ratnesh.pasi@iitgn.ac.in
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

Abstract

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 insight. 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.*

Keywords

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 *Proceedings of 13th International Conference on Data Science (CODS 2025).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

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.

*Both authors contributed equally to this research.

[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.

CODS 2025, IISER, Pune, India

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXX.XXXXXXX

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 pass@1 value of 0.79, followed by Qwen3-32B with pass@1 value of 0.78. Common syntactic error types 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 MIMIC-SQL [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 [3], 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 Code-Contests [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 LogicNLG (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

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 [1]	✗	✓	✗	✗	✗
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)	✓	✓	✓

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.

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*. 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] https://huggingface.co/spaces/SustainabilityLabIITGN/VayuChat

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).

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

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

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).

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

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

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).

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

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

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. 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?*

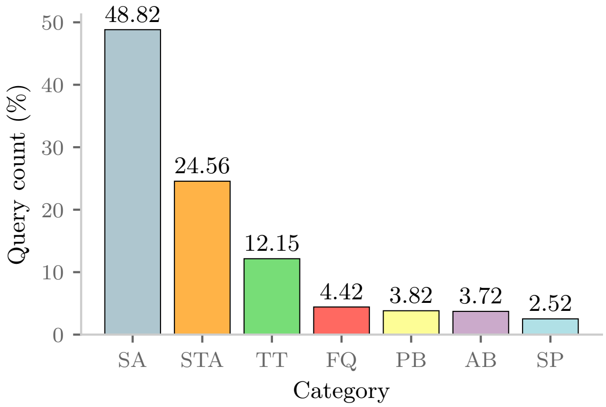


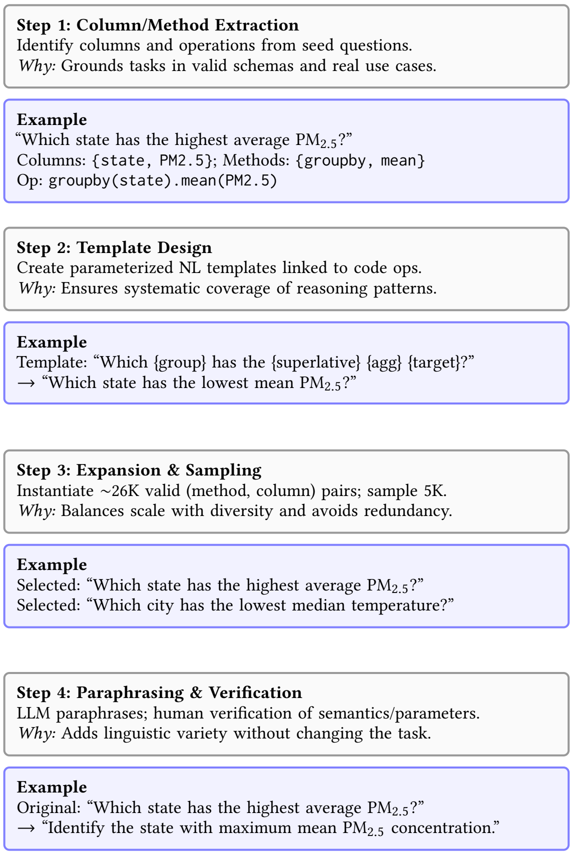
Figure 1: Distribution of the 5,000 benchmark queries across categories. Spatial, spatio-temporal, and temporal queries dominate due to richer variability in station- and city-level data, while funding- and population-linked queries are fewer, reflecting their narrower schema coverage.

(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 handcrafted 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.



Step 1: Column/Method Extraction
Identify columns and operations from seed questions.
Why: Grounds tasks in valid schemas and real use cases.

Example
"Which state has the highest average $PM_{2.5}$?"
Columns: {`state, PM2.5`}; Methods: {`groupby, mean`}
Op: `groupby(state).mean(PM2.5)`

Step 2: Template Design
Create parameterized NL templates linked to code ops.
Why: Ensures systematic coverage of reasoning patterns.

Example
Template: "Which {group} has the {superlative} {agg} {target}?"
→ "Which state has the lowest mean $PM_{2.5}$?"

Step 3: Expansion & Sampling
Instantiate ~26K valid (method, column) pairs; sample 5K.
Why: Balances scale with diversity and avoids redundancy.

Example
Selected: "Which state has the highest average $PM_{2.5}$?"
Selected: "Which city has the lowest median temperature?"

Step 4: Paraphrasing & Verification
LLM paraphrases; human verification of semantics/parameters.
Why: Adds linguistic variety without changing the task.

Example
Original: "Which state has the highest average $PM_{2.5}$?"
→ "Identify the state with maximum mean $PM_{2.5}$ concentration."

Figure 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.

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:

- A fixed function signature: `get_response(data, states_data, ncap_funding_data)`.
- Output restricted to a single scalar value (no DataFrames, tuples, or plots).
- Explicit imports and code-only responses, wrapped in `<code>...</code>`.
- 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. Pilot runs showed diminishing returns beyond $n = 5$, making this choice a practical balance between diversity and efficiency.
We used the following evaluation metrics:

- **pass@k:** Fraction of queries where at least one of the top-k completions 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 = $\frac{1}{N}\sum_{i=1}^{N}\frac{c_i}{n}$.

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 coding pretraining.
- **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.** GPT-OSS-20B excels in funding related queries (FQ) despite weaker structured performance, 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(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.

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).

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 surge 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 $\leq$ 0.12. Interestingly, the specialized *Qwen3-Coder-30B* attains top accuracy (exec@1 = 0.99) but does not eliminate column errors completely, 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

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	0.0	0.35	0.0	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 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.

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

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.

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

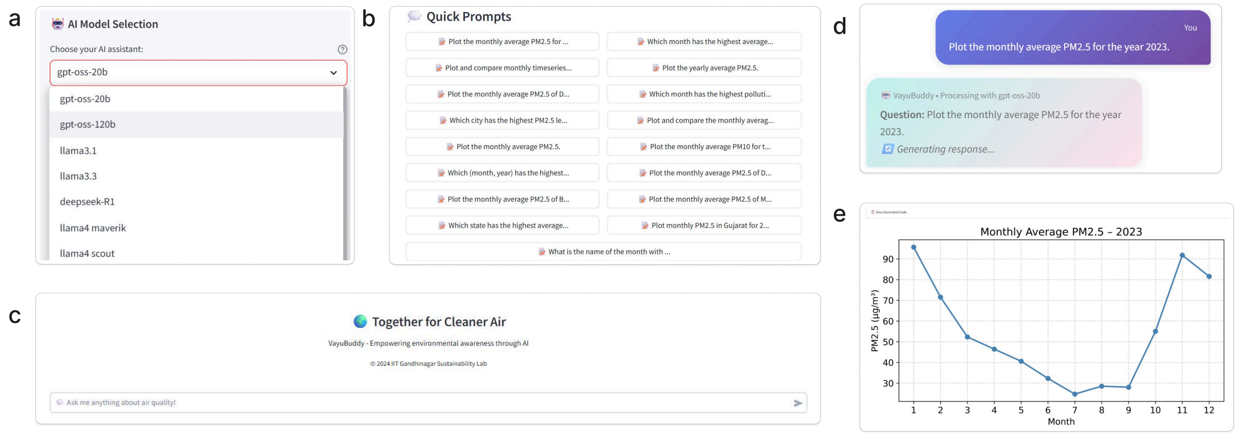
VayuChat is deployed as a web application on Hugging Face Spaces[4], offering an intuitive interface for interactive querying of Indian air quality data (Figure 3).

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.

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—are cached for efficiency. A verification layer checks

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


a
AI Model Selection
Choose your AI assistant:
gpt-oss-20b
gpt-oss-20b
gpt-oss-120b
llama3.1
llama3.3
deepseek-R1
llama4 maverik
llama4 scout
b
Quick Prompts
Plot the monthly average PM2.5 for ...
Which month has the highest average...
Plot and compare monthly timeseries...
Plot the yearly average PM2.5.
Plot the monthly average PM2.5 of D...
Which month has the highest polluti...
Which city has the highest PM2.5 le...
Plot and compare the monthly averag...
Plot the monthly average PM2.5.
Plot the monthly average PM10 for t...
Which (month, year) has the highest...
Plot the monthly average PM2.5 of D...
Plot the monthly average PM2.5 of B...
Plot the monthly average PM2.5 of M...
Which state has the highest average...
Plot monthly PM2.5 in Gujarat for 2...
What is the name of the month with ...
c
Together for Cleaner Air
VayuBuddy - Empowering environmental awareness through AI
© 2024 IIT Gandhinagar Sustainability Lab
Ask me anything about air quality!
d
e
View Generated Code
Monthly Average PM2.5 – 2023
PM2.5 (µg/m³)
90
80
70
60
50
40
30
1
2
3
4
5
6
7
8
9
10
11
12
Month

You

Plot the monthly average PM2.5 for the year 2023.

VayuBuddy • Processing with gpt-oss-20b

Question: Plot the monthly average PM2.5 for the year 2023.

Generating response...

Figure 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, statistics, and explanations.

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 PM2.5 in Delhi over the past year), comparative analysis (e.g., monthly PM2.5 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.

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

LLM (wrong):

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

True code (correct):

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

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

6 Limitations and Future Work

VayuBench and VayuChat currently rely on CPCB/DPCC 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 [2], Particle [33]) and open-weight LLMs (LLaMA-3, Mistral) will enhance robustness and efficiency for deployment.

Q (SA): On Mar 31, 2023, which state had the 3rd-lowest 25th percentile for PM_{10}?

LLM Output: 31.3582
Correct Ans.: Haryana

LLM (wrong):

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

True code (correct):

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

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

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 (ML) and Air Quality (AQ) communities. This work introduces a large-scale benchmark of over 5,000 diverse, templated, and paraphrased questions across seven categories, enabling rigorous assessment of code-generating LLMs on real, structured air-quality datasets, along with a reproducible evaluation framework using ground-truth code annotations and pass@k metrics, advancing best practices for LLM benchmarking in code-based 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] Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

[2] Anyscale (Ray project). Ray: Scale machine learning & ai computing. https://www.ray.io/, 2025. Accessed: 2025-08-18.

[3] Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. Program synthesis with large language models. In *ICLR* (2021).

[4] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020.

[5] Carslaw, D. C., and Ropkins, K. openair — an r package for air quality data analysis. *Environmental Modelling Software 27–28*, 0 (2012), 52–61.

[6] Central Pollution Control Board. Cpcb - overview and functions. https://cpcb.nic.in/, 2025.

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

[8] Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021.

[9] Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021.

[10] Chen, W., Chen, J., Su, Y., Chen, Z., and Wang, W. Y. Logical natural language generation from open-domain tables, 2020.

[11] Clarity Movement. How open access air pollution data is paving the way for greater air quality awareness, Feb. 2022. Accessed: 2025-05-15.

[12] Contalbo, M. L., Pederzoli, S., Buono, F. D., Valeria, V., Guerra, F., and Paganelli, M. GRI-QA: a comprehensive benchmark for table question answering over environmental data. In *Findings of the Association for Computational Linguistics: ACL 2025* (Vienna, Austria, July 2025), W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, Eds., Association for Computational Linguistics, pp. 15764–15779.

[13] CPCB. National air quality index real-time data, 2025. Accessed: 2025-05-15.

[14] Dalbah, Y., Worring, M., and Hsu, Y.-C. Veli: Unsupervised method and unified benchmark for low-cost air quality sensor correction, 2025.

[15] Eliseeva, A., Kovrigin, A., Kholkin, I., Bogomolov, E., and Zharov, Y. Envbench: A benchmark for automated environment setup, 2025.

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

[17] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938* (2021).

[18] Herzig, J., Nowak, P. K., Müller, T., Piccinno, F., and Eisenschlos, J. M. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349* (2020).

[19] Hettige, K. H., Ji, J., Xiang, S., Long, C., Cong, G., and Wang, J. Airphynet: Harnessing physics-guided neural networks for air quality prediction, 2024. Accepted by ICLR 2024.

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

[21] Lai, J., Zhang, J., Liu, J., Li, J., Lu, X., and Guo, S. Spider: Any-to-many multimodal llm, 2025.

[22] Lai, Y., Li, C., Wang, Y., Zhang, T., Zhong, R., Zettlemoyer, L., tau Yih, S. W., Fried, D., Wang, S., and Yu, T. Ds-1000: A natural and reliable benchmark for data science code generation, 2022.

[23] Lee, G., Hwang, H., Bae, S., Kwon, Y., Shin, W., Yang, S., Seo, M., Kim, J.-Y., and Choi, E. Ehrsql: A practical text-to-sql benchmark for electronic health records, 2023.

[24] Li, J., Hui, B., Qu, G., Yang, J., Li, B., Li, B., Wang, B., Qin, B., Cao, R., Geng, R., Huo, N., Zhou, X., Ma, C., Li, G., Chang, K. C. C., Huang, F., Cheng, R., and Li, Y. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls, 2023.

[25] Li, Y., Choi, D., Gerstenberg, M., Humayoun, U., Jose, A. R., Kohli, P., Koo, J., Lee, Y., Li, L., Li, W., et al. Competition-level code generation with alphacode. *Science 378*, 6624 (2022), 1092–1097.

[26] Lu, S., Guo, D., Ren, S., Huang, J., Svyatkovskiy, A., Blanco, A., Clement, C., Drain, D., Jiang, D., Tang, D., Li, G., Zhou, L., Shou, L., Zhou, L., Tufano, M., Gong, M., Zhou, M., Duan, N., Sundaresan, N., Deng, S. K., Fu, S., and Liu, S. Codexglue: A machine learning benchmark dataset for code understanding and generation, 2021.

[27] Moosavi, N. S., Fichtel, L., Pado, S., and Gurevych, I. Scigen: A dataset for scientific table-to-text generation. In *ACL* (2021), pp. 4376–4389.

[28] Nan, L., Yu, J., Zhang, X., and Radev, D. Fetaqa: Free-form table question answering. In *ACL* (2022), pp. 1020–1034.

[29] OpenAQ Project. Openaq: Open air quality data platform. https://openaq.org/, 2024.

[30] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022.

[31] Pandey, A., Brauer, M., Cropper, M. L., Balakrishnan, K., Mathur, P., Dey, S., Turkgulu, B., Kumar, G. A., Khare, M., Beig, G., et al. 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] Parikh, A. P., Wang, X., Gehrmann, S., Faruqui, M., Dhingra, B., Yang, D., and Das, D. Totto: A controlled table-to-text generation dataset, 2020.

[33] Particle. Particle: An integrated iot platform-as-a-service. https://www.particle.io/, 2025. Accessed: 2025-08-18.

[34] Pasupat, P., and Liang, P. Compositional semantic parsing on semi-structured tables. In *ACL* (2015), pp. 1470–1480.

[35] Puri, R., Kung, D., et al. Codenet: A large-scale ai for code dataset for learning a diversity of coding tasks. *arXiv preprint arXiv:2105.12655* (2021).

[36] Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Bhosale, S., Wang, X. L., Derczynski, L., Chrzanowski, M., Scialom, T., et al. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761* (2023).

[37] Vito, S. D., Esposito, E., Salvato, M., Popoola, O., Formisano, F., Jones, R., and Francia, G. D. 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.

[38] Wang, P., Shi, T., and Reddy, C. K. 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)* (Online, 2020), Association for Computational Linguistics, ACL.

[39] Wei, X., Cui, X., Cheng, N., Wang, X., Zhang, X., Huang, S., Xie, P., Xu, J., Chen, Y., Zhang, M., Jiang, Y., and Han, W. Zero-shot information extraction via chatting with chatgpt, 2023.

[40] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models, 2023.

[41] Yu, T., Zhang, R., Yasunaga, M., Tan, Y. C., Lin, X. V., Li, S., Er, H., Li, I., Pang, B., Chen, X., Shi, T., Li, Z., Jiang, Y., Yasunaga, M., and Radev, D. Sparc: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), pp. 4511–4523.

[42] Zhong, V., Xiong, C., and Socher, R. 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)* (2017), pp. 231–240.

[43] Zuo, J., Li, W., Baldo, M., and Hacid, H. Unleashing realistic air quality forecasting: Introducing the ready-to-use purpleairsf dataset, 2023. Accepted by ACM SIGSPATIAL 2023.