Data Infrastructure and Management in Context of AI

Fall 2026 | Tuesdays 5:30- 7:30 PM | October 20 to December 1, 2026

> *This course is designed for mixed technical backgrounds. Students with no prior coding experience are fully welcome but should expect to invest additional time outside of class on the pre-course primer before Session 1.*

Course Description

This course focuses on developing the **engine room skills** required to design and manage modern data systems. It aims to build foundational capabilities in data infrastructure, pipeline design, and architectural thinking as durable skills that support analytical work and **AI-enabled** systems across contexts. The course emphasizes infrastructure fundamentals through engagement with a modern technical stack including SQL, Python, APIs, and cloud storage, focusing on how data is orchestrated from raw sources to downstream interfaces. Core topics examine how data moves through systems, how readiness and reliability are established, and how design choices shape system behavior over time. The course further develops literacy by examining how traditional analytical workflows connect to emerging system architectures, including large language model-based interfaces. It culminates in the design of a system-ready data pipeline that demonstrates the ability to translate complex data into a reliable, interpretable interface for end users.

Academic Integrity and AI Use Policy

Honor Code

All students are expected to adhere to Columbia University's academic integrity policies and the GSAPP Honor Code. Submitted work must be your own. Misrepresenting the source or authorship of your work is a violation of the Honor Code and may result in academic consequences. Full GSAPP policy available at gsapp.columbia.edu.

AI Use in This Course

This course engages directly with AI tools as infrastructure, not as shortcuts. The goal is to understand how these systems work from the ground up.

- AI coding assistants (ChatGPT, Claude, GitHub Copilot, etc.) are permitted and encouraged for debugging, exploring syntax, and learning. The standard is simple: you must be able to explain every line of code you submit. If you cannot explain it, it is not yours.
- Submitting AI-generated code or written work you do not understand is a violation of academic integrity, regardless of how it was produced.
- Written reflections, architecture documentation, and peer review forms must reflect your own thinking. Use AI to sharpen your ideas, not to replace them.
- When in doubt, document it. If you used an AI tool substantially in your work, note how in your submission. Transparency is not penalized; undisclosed over-reliance is.

Learning Outcomes

By the end of this course, students will be able to:

- Connect to a live, open-source data feed and ingest it into a structured storage layer
- Write SQL queries and basic Python scripts to clean and transform raw data
- Apply automated validation checks to establish data readiness before downstream use
- Prepare structured data for consumption by a large language model
- Orchestrate a repeatable, schedulable data pipeline using Mage
- Build and deploy a Streamlit application that allows non-technical users to ask plain-English questions of their data
- Articulate the architectural decisions behind a working data product and defend their tradeoffs

Session Schedule

Session	Date	Topic	Lab Focus
01	Oct 20	**The Engine Room: Infrastructure and Stack Orientation**	Colab setup, DuckDB first query, data lifecycle map
02	Oct 27	**Relational Thinking + Data in Motion**	SQL queries + Python ingestion script in one notebook
03	Nov 3	**Live Data + Trusting the Data**	API ingestion into DuckDB + validation checks
04	Nov 10	**Making Data AI-Ready + Pipeline Architecture with Mage**	LlamaIndex query + Mage pipeline build
05	Nov 17	**The Interface: Streamlit + LLM Q&A**	Deploy a working Streamlit app over your data
06	Nov 24	**Project Workshop + Peer Review**	Guided build time, structured peer feedback
07	Dec 1	**System-Ready: Final Presentations**	Live demos + architecture critique

Session Details

Session 1 | Tuesday, October 20

The Engine Room: Infrastructure and Stack Orientation

We begin by asking why infrastructure matters, not as plumbing, but as the architecture of trust. This session orients students to the full pipeline arc, the tools they will use throughout the course, and the question that drives the final project: what does it take to turn raw data into something a non-technical user can interrogate? We set up environments and run a first query together.

- The data lifecycle: ingestion → validation → transformation → AI interface
- Why AI systems are only as reliable as the infrastructure beneath them
- Tour of the course stack
- The final project arc: open data → pipeline → scheduled reports → AI-powered UI
- Discussion: What data problem would you want to make queryable?

Lab: *Set up tech stack. Connect to a sample dataset. Run your first SQL query. Make three observations about the data via SQL*

Session 2 | Tuesday, October 27

Relational Thinking + Data in Motion

This combined session covers two foundational skills in one arc: how to think about data structure (SQL and the relational model) and how to move and reshape data using Python. Rather than treating these as separate topics, we frame them together, schema design informs how you write transformation scripts, and transformation scripts reveal the limits of your schema. By end of session, students have an ingestion notebook that reads raw data and loads it into a structured DuckDB table.

- Tables, keys, and relationships: the relational mental model
- Core SQL: SELECT, JOIN, GROUP BY, aggregation
- Reading and cleaning data with pandas: nulls, types, deduplication
- Writing data from Python into DuckDB
- Schema design as an architectural decision: what structure does your data need?

Lab: *Write a Python script in Colab that reads a messy CSV dataset, applies three cleaning steps, and loads the result into a DuckDB table. Query the table to verify the output.*

Students should complete SQLBolt Lessons 1–12 before this session (see pre-course primer).

Session 3 | Tuesday, November 3

Live Data + Trusting the Data

This combined session connects students to live, external data for the first time and immediately asks: how do you know you can trust it? We introduce REST APIs as the mechanism for pulling real-world data, then build validation logic directly into the ingestion notebook. The goal is to develop the instinct that ingestion and quality checking are inseparable

- REST API anatomy: endpoints, authentication, pagination, rate limits
- Fetching live data in Python with the requests library
- Storing API responses in DuckDB
- Data quality dimensions: completeness, accuracy, consistency, freshness
- Writing validation assertions in Python: what does “ready” mean for your data?
- Designing for failure: what happens when the data doesn’t pass?

Lab: *Connect to a live public API (options provided or use your project dataset). Fetch at least two pages of data, load into DuckDB, and write five validation checks that must pass before the data is usable downstream.*

Project dataset selection due by before next session.

Session 4 | Tuesday, November 10

Making Data AI-Ready

With clean, validated data in hand, we now ask a different question: what does a language model need from your data to answer questions well? This session introduces Retrieval-Augmented Generation (RAG): the architecture that grounds LLM responses in real data rather than general knowledge, connects a DuckDB dataset to an LLM API for the first time. Students run their first plain-English query against their own data.

- How LLMs process information: context windows and their limits
- What is RAG? Why grounding matters for reliability and trust
- Chunking strategies: how to break data into pieces an LLM can use
- Connecting your data to an LLM in minimal code
- The prompt as a data contract: what structure your LLM expects
- Testing your first query: what works, what doesn’t, and why

Lab: *Take your validated DuckDB dataset and connect it to an LLM. Ask five plain-English questions. Document which answers are good, which are weak, and what the data or chunking strategy would need to change to improve them.*

Session 5 | Tuesday, November 17

The Interface: AI Query Layer

The final component of the system-ready pipeline is the interface itself. This session builds the query interface: a text input, a response panel showing grounded answers, and a data view showing what the system is drawing from. By the end of class, every student has a running local app.

- What is Streamlit and why it works for data products
- Building a Streamlit app: inputs, outputs, and layout
- Connecting Streamlit to your pipeline
- Displaying data and answers side by side
- Trust design: how to show users what the system knows and doesn't know

Lab: *Rebuild your ingestion + validation workflow as a Mage pipeline with at least three blocks (load, validate, transform). Schedule it to run daily. Verify the run log shows a successful execution.*

Final project proposal due by end of this session: one paragraph describing your dataset, your pipeline plan, and the question your Streamlit app will help users answer.

Session 6 | Tuesday, November 24

Project Workshop and Peer Review

This session is structured protected build time. The first half is an open workshop where the instructor circulates, and students work on their final pipelines with direct support available. The second half is a structured peer review: students demo their current Streamlit app to a partner, who asks three real questions and gives structured feedback on what the system handles well and where it breaks down. The goal is to surface problems with a week still to fix them.

- Open workshop: pipeline refinement, validation debugging, interface polish
- Common failure modes: what goes wrong in RAG pipelines and how to fix it
- Structured peer demo: three questions, documented feedback
- Architecture review checklist: is your pipeline complete end-to-end?
- Presentation preparation: how to explain a technical system to a non-technical audience

Lab: *Complete the peer review form for your partner's app. Submit your own updated pipeline with at least one improvement based on the feedback received.*

Final deliverable due Sunday, November 29 by midnight. Submit the final URL.

Session 7 | Tuesday, December 1

System-Ready: Final Project Presentations

The culminating session is a live demonstration and critique. Each student presents their complete data system, from live API ingestion through Mage-orchestrated pipeline, validation, AI-readiness layer, and Streamlit interface, to the class. Presentations focus on what the system makes possible, the architectural decisions made along the way, and an honest account of what it still can't do.

- Live demo of the complete pipeline: open data → Mage → DuckDB → LlamaIndex → Streamlit
- 5-minute architecture walkthrough: three decisions you made and why
- One thing the system handles well; one thing it doesn't
- Peer and instructor Q&A: real questions asked through the Streamlit interface live

Lab: *Present your system-ready pipeline. Graded on functionality, design integrity, quality architecture, and clarity of presentation.*

Presentations are 8 minutes each with 4 minutes of Q&A. Order will be shared the week before.

Assessment and Grading

Component	Weight	Description
Participation & In-Class Engagement	**10%**	Evaluated across quality of contribution to discussion, in-class exercises, and completion and depth of the Session 6 peer review
Lab Exercises (Sessions 1–5)	**30%**	6% per lab. Graded on completion and design.
Final Project Deliverable	**40%**	Functional pipeline + deployed Streamlit app submitted by Nov 29 with documentation
Final Presentation (Dec 1)	**20%**	Live demo and architecture walkthrough; real Q&A through the Streamlit interface

Readings and Resources

Resources are organized in two tiers: a pre-course primer to establish a shared foundation before Session 1, and per-session materials that prepare you for each week's concepts and lab. Students with prior Python or SQL experience may move quickly through foundational items. Students with no coding background should treat the pre-course primer as required, not optional, it is the difference between keeping up and falling behind in Sessions 2 and 3.

Pre-Course Primer

Complete before October 13. Estimated time: 3 - 5 hours total depending on your background.

Data Infrastructure Orientation (everyone)

- Fundamentals of Data Engineering, Ch. 1: "Data Engineering Described", Reis & Housley (O'Reilly, 2022). Sets the conceptual frame for the entire course. Available via O'Reilly Online or library access.
- What Is a Data Pipeline? Hazel Virdó, Towards Data Science. A concise plain-English overview of pipeline stages and why they matter.

Python Orientation (if new to Python, required for beginners)

- Python for Everybody, Ch. 1–5, Dr. Chuck Severance (py4e.com). Variables, conditionals, loops, functions, and files. Takes 2–3 hours. Do not skip this if you have never written a Python script.
- Google's Python Class, developers.google.com/edu/python. A faster-paced alternative if you have some logic background but no Python experience.

SQL Orientation (if new to SQL, required for beginners)

- SQLBolt, sqlbolt.com, interactive, browser based. Complete Lessons 1–12. Takes roughly 90 minutes and requires no setup. Do this before Session 2.
- Mode SQL Tutorial, mode.com/sql-tutorial. A good alternative with a more analytical framing.