ALT/FNDATA ACADEMY · FACILITATOR GUIDE

Facilitator guide: Module 12, Data engineering and the API

A teacher-facing companion to Module 12. The module is the full lesson; this guide adds the facilitation layer: a preparation checklist, timing cues, guidance for the discussion questions, and notes on what to emphasize and where students get stuck. Slides and a printable version are below.

At a glance

• **Length:** 90 minutes. A 60 minute and a 120 minute variant are noted under Teaching notes.

• **Level:** undergraduates or graduate students who have taken an introductory programming or data science course. The core session assumes no prior API experience; the optional extension assumes basic Python familiarity.

• **Access:** the no-code sandbox at sandbox.altfndata.com opens the session, and it carries the core 70 minutes on its own. The optional extension has a substantial API and code component and requires an instructor class API key requested from info@altfndata.com, plus the downloadable Python client or tutorials notebook at docs.altfndata.com.

• **Goal of the session:** students leave able to describe the shape of the production query endpoint, explain why it paginates on offset, and, if time allows, run a small client-side ETL that pulls one brand's rows and computes an aggregate themselves.

Before class

• Register a sandbox account and open the table list and schema tab once beforehand, so you can narrate discovery fluently.

• If running the optional code extension, request a class API key from info@altfndata.com well ahead of class, and test the Python client (altfndata_client.py) end to end using the worked pagination example.

• Read through the query endpoint's request body and response envelope in the documentation until you can describe fields, filters, sort, limit, and offset without reading off a slide.

• Decide before class whether you will run the optional extension. This determines whether you need the API key and reshapes the 55 to 70 minute segment into a live-code demo instead of a documentation walkthrough.

• Prepare a printed or displayed copy of the first-page and second-page request body examples for students following along without their own device.

Timed agenda with cues

Time	Segment	What to do	Watch for
0 to 10	Introduce the session's frame	Preview understanding a data provider's contract before building on it, and name the two discovery endpoints and the single query endpoint every table shares.	Students expecting a coding session from minute one. Set the expectation that the first half is documentation and structure, not code.

Time	Segment	What to do	Watch for
10 to 25	Sandbox orientation and discovery	Students use the table list and schema tab to answer what tables exist and what fields a given table exposes.	Students trying to query a field that is not documented. Redirect them to the schema tab before they guess.
25 to 40	Guided walkthrough, query endpoint shape	Walk the request body's four parts (fields, filters, sort, limit, offset) and the response envelope (table, result_count, data) using the worked example.	Students assuming the response includes a pre-computed total or average. Point out plainly that data holds only rows.
40 to 55	Guided walkthrough, pagination	Work through the offset example: a brand with 3,400 matching rows needs four requests, each advancing offset by the rows already retrieved.	Students confusing offset with a page number. Show the second-page request body side by side with the first.
55 to 70	Optional code extension	Using the class API key and the Python client, live-run the small ETL: pull one brand's rows across all pages and compute a client-side aggregate.	Pagination loop bugs, especially forgetting to advance offset. Narrate the loop's termination condition explicitly as you run it.
70 to 85	Small-group exercise	Groups design the query body for a specific question about a chosen brand, on paper or in the sandbox, and run it if doing the code extension.	Groups reaching for a filter operator that does not exist. Have the documented operator list visible for reference.
85 to 90	Wrap-up and homework	Restate the one-sentence takeaway and hand out the homework.	Leave 2 minutes for the homework logistics, not zero.

Guidance for the discussion questions

Use these as the "what to listen for" behind each question in the module. They are talking points, not a graded key.

1. **Why discovery should precede querying.** Listen for: guessing at a table or field name either errors outright or, worse, silently returns an empty result that looks like "no data" instead of "wrong field name." Discovery removes that ambiguity before any real query gets built.

2. **What breaks without pagination, and how offset guards against it.** Listen for: a client assuming a single request returns an unlimited number of rows would either get a silently truncated result or hit the roughly 1,000-row ceiling and mistake it for the whole answer. Offset-based pagination makes the client explicitly request each subsequent slice, and comparing the returned row count to the limit requested is what signals the last page.

3. **Rows versus server-side aggregates, and the tradeoff.** Listen for: returning rows keeps the API simple and general across every caller's use case, but shifts the work, and the correctness burden, onto each client to compute its own aggregate. The tradeoff is a simpler, more general API against more responsibility for the client.

4. **What the 11 filter operators can and cannot answer.** Listen for: eq, like, in, and the comparison and null operators cover most filter-and-sort questions, such as a brand, a date range, a status, or a missing estimate. Anything requiring a value computed across rows, such as an aggregate or a join across tables, is out of scope and must happen client-side after retrieval.

5. **Manually issued keys and class provisioning.** Listen for: manual issuance is a deliberate access-control choice. For a course, it implies one shared instructor-held class key rather than self-serve student keys, and instructors should request that key from info@altfndata.com well before the session.

6. **A pagination bug that never advances the offset.** Listen for: the client would re-request and receive the same first page indefinitely, either looping forever or accumulating duplicate rows. The tell in the response is that result_count and the returned rows never change between requests, which is verifiable by comparing two consecutive pages.

7. **Checking the schema before every production query.** Listen for: a table's schema can change over time, such as a renamed or added field, and a pipeline that assumes a fixed schema will fail silently or loudly when it changes. Checking the schema endpoint before each query, or on a schedule, makes a pipeline more resilient to that drift.

Teaching notes

• **Most common misconception:** students assume the API can compute an average or count server-side. Point at the response envelope showing only data rows, then run the client-side aggregate live so the correction is visible, not just stated.

• **Second misconception:** confusing offset-based pagination with page-number pagination. Show the second-page request body next to the first and have students identify exactly which value changed and why.

• **60 minute variant:** skip the optional code extension entirely and compress the small-group exercise to designing a query body only, no running it. Stay in the no-code discovery-and-documentation path throughout.

• **120 minute variant:** have each small group actually run the Python client extension themselves, using the class key, instead of only watching the instructor demo, then compare aggregates across groups' chosen brands.

• **If the sandbox or API is slow or blocked on the room network:** fall back to the pre-captured worked request and response examples, and use the pre-run ETL output you saved in preparation rather than live-calling the API.

For the full lesson content, queries, and homework, see Module 12. Questions or a class API key: info@altfndata.com.