ALT/FNDATA ACADEMY · STUDENT STUDY SHEET

Study sheet: Module 12, data engineering and the API

This session moves students from querying data in a browser to understanding how a production data API is structured and how a small pipeline is built on top of it. Students start in the no-code sandbox, using its discovery views to see what a table catalog and a schema endpoint actually expose, then move to the documented shape of the production API itself: a single query endpoint that every table shares, a small set of filter operators, and pagination on an offset rather than a page number. The session closes with an optional, hands-on extension in which students use a class API key and a reusable Python client to pull one brand's transaction rows, store them, and compute an aggregate themselves, since the API is deliberately built to return rows rather than server-side aggregates. The throughline is a data engineering lesson as much as a finance one: understanding the contract a data provider offers is the first step in building anything reliable on top of it.

What you should be able to do

1. Use the sandbox's discovery views to identify which tables exist and which fields a given table exposes, and explain why discovery should precede any query design.
2. Describe the shape of the production API's query endpoint, including the request method, the required header, and the four parts of the request body.
3. Explain why the API paginates on an offset rather than a page number, and describe how a client would retrieve a full result set larger than one page.
4. Distinguish a query that filters and sorts rows from a query that aggregates, and explain why this API returns rows and leaves aggregation to the client.
5. (Extension) Use the reusable Python client to authenticate a request, retrieve paginated results for a single brand, and compute a simple aggregate, such as a count or an average, from the retrieved rows.
6. Identify the operational limits of the API relevant to a data engineering context, including the page size ceiling and the fact that API keys are issued manually by the team rather than self-served.

Key terms

- **Discovery endpoint:** an API endpoint whose purpose is to describe what is available, here GET /v1/tables and GET /v1/tables/{name}/schema, rather than to return transaction data.
- **Request body:** the JSON object sent with a POST request, here containing fields, filters, sort, limit, and offset.
- **Response envelope:** the wrapping structure of an API response, here table, result_count, and data, that surrounds the actual rows returned.
- **Pagination:** the practice of splitting a large result set across multiple requests, here controlled by limit and offset rather than a page number.
- **Offset:** the number of rows to skip before starting to return results, used here to advance through a paginated result set.
- **Filter operator:** a documented comparison a query can apply to a field, such as eq, like, in, gt, gte, lt, lte, or is_null.

- **Client-side aggregation:** computing a summary statistic, such as a count or an average, from raw rows already retrieved, rather than asking the server to compute it.
- **API key:** a credential passed in the X-API-Key header that authenticates a request; in this course, issued manually to the instructor as a shared class key.

Full lesson, more queries, and the homework are in Module 12 at academy.altfndata.com. Questions: info@altfndata.com.