ALT/FNDATA ACADEMY · STUDENT STUDY SHEET

Study sheet: Module 6, finTech and data products

This session treats ALT/FNDATA itself as the teaching case: how does a raw, messy, physical-world dataset such as auction and resale transaction records become a commercial data product with a schema, an access tier structure, and a REST API. Students examine the three access surfaces a real data vendor maintains (a no-code sandbox, documentation with a client library, and a production API) and discuss the product and business decisions behind each one, including why self-serve access and manually issued API keys serve different customer segments. The session closes with students sketching their own access-tier design for a hypothetical alternative data product, using ALT/FNDATA's structure as a reference point rather than a template to copy exactly.

What you should be able to do

1. Describe the three access surfaces of a data product (no-code sandbox, documentation and client library, production API) and the distinct customer need each one serves.
2. Explain the tradeoffs between self-serve access and manually issued credentials, using API key provisioning as a concrete example.
3. Read and interpret a REST API's request and response shape, including endpoint, headers, filters, and pagination.
4. Identify what a data dictionary and coverage browser contribute to a data product's usability, separate from the raw data itself.
5. Design a basic access-tier structure for a hypothetical data product, justifying which tier gets self-serve access and which requires manual approval.
6. (Extension) Issue a working API request using the reusable Python client and interpret the response envelope.

Key terms

- **Access tier:** a level of product access defined by what a customer can request and how much oversight is required to unlock it.
- **Endpoint:** a specific URL path in an API that performs one operation, such as running a table query or listing available tables.
- **Response envelope:** the wrapper object returned around the actual rows, here table, result_count, and data.
- **Pagination:** retrieving a large result set in fixed-size pages using an offset, rather than requesting everything at once.
- **Schema:** the list of fields and their types available on a given table, returned by the schema endpoint.
- **Self-serve:** an access flow a customer completes without a human on the vendor's side approving each step.
- **Client library:** reusable code, here a Python client, that wraps the raw API calls a developer would otherwise write by hand.
- **Filter object:** a single condition inside an API request body, expressed as a field, an operator, and a value.

Core sandbox query

The query the module is built around. Run it, then change the brand or category and compare.

```
SELECT item_title, sale_date, usd_price_decimal, vendor
FROM all_fine_art_data
WHERE status = 'sold'
ORDER BY sale_date DESC
LIMIT 25;
```

Common mistakes to avoid

1. Assuming result_count reflects the total number of matching rows rather than just the rows returned on the current page.
2. Forgetting to advance offset between requests, so a script silently re-reads the first page every time.
3. Treating the sandbox's auto-approval as evidence that the production API is equally open, when production keys are manually issued.
4. Featuring or promising the AI natural language search capability, which is still in development and not part of the current product.
5. Confusing the fields listed in one query's request body with the full schema of a table, and missing fields that exist but were not requested.
6. Designing a hypothetical access tier without a clear reason why it is self-serve rather than manually approved, beyond habit.

Full lesson, more queries, and the homework are in Module 6 at academy.altfndata.com. Questions: info@altfndata.com.