ALT/FNDATA ACADEMY · MODULE 2

Data science and SQL

This session is a hands-on SQL workshop built on a real institutional dataset rather than a toy database. Students practice filtering, grouping, and aggregating transaction-level records to build a simple demand index, one of the core analytical patterns used by working data analysts.

90 minutes · sandbox.altfndata.com

SESSION AT A GLANCE

What today looks like

- Length: 90 minutes. This is the longest session in the sequence, since it carries three guided queries plus independent practice.
- Level: undergraduate students who have completed an introductory SQL unit, or graduate students in a data analytics or business intelligence program. Comfort with SELECT, WHERE, GROUP BY, ORDER BY, and basic aggregate functions is assumed.
- Access: the no-code sandbox at sandbox.altfndata.com, self-registered and auto-approved. An optional code extension shows the same query issued against the production API with a Python client, for students who requested a class API key in advance from info@altfndata.com.
- Goal of the session: students leave able to write a filtered, grouped, and ordered query against a real multi-million-row dataset, build a quarterly demand index, compute a pricing power ratio, and explain a data quality issue (unsold lots) they must not discard.

OBJECTIVES

By the end of class students can

- Write SELECT statements with WHERE, GROUP BY, and ORDER BY clauses against a real multi-million-row dataset.
- Construct an aggregate metric (a brand-level demand index) from raw transaction records using quarterly bucketing.
- Compute a ratio metric (pricing power, defined as median realized price divided by high estimate) and explain what values above or below 1.0 mean.
- Identify and handle a data quality issue, specifically unsold lots, without discarding rows that matter to the analysis.
- Translate a SQL query result into a short written interpretation suitable for a non-technical stakeholder.
- (Extension) Reproduce a sandbox query as a POST request against a REST API and compare the two workflows.

AGENDA

How the session runs

- 0 to 10 min: Recap of SQL fundamentals and introduction to the dataset's shape: what a transaction record contains and which fields are documented (designer, model, item_title, sale_date, usd_price_decimal, sale_estimates_high_usd_price, status, vendor, stock_ticker).
- 10 to 25 min: Guided query 1, filtering and sorting a single brand's sold lots.
- 25 to 45 min: Guided query 2, building the quarterly demand index with GROUP BY and date bucketing.
- 45 to 60 min: Guided query 3, computing pricing power as a ratio metric and discussing the sell-through concept.
- 60 to 75 min: Independent practice, students adapt the demand index query to a brand of their choice.
- 75 to 85 min: Optional extension demo, instructor shows the same demand index query issued via the production API with the Python client.
- 85 to 90 min: Wrap-up and homework assignment.

LIVE DEMO

The in-class walkthrough

- Open sandbox.altfndata.com, sign in, and select the jewelry data table from the SQL editor dropdown.
- Run the first guided query below and review the columns returned: designer, item_title, sale_date, usd_price_decimal.
- Introduce the concept of date bucketing by asking students how they would group these rows into quarters using only SQL functions available in the editor.
- Run the second guided query, which buckets sold lots by quarter and computes a count and average realized price for each quarter, and display the result as a line chart using the pre-built charts tab.
- Run the third guided query, which computes the ratio of median realized price to median high estimate for a single brand, and explain that a result above 1.0 means buyers paid over the pre-sale estimate.
- Ask students to change the designer filter in query 3 to a second brand and compare the two ratios.

LIVE DEMO

The core query

- Run this in the sandbox SQL editor:

```
SELECT designer, item_title, sale_date, usd_price_decimal
FROM all_jewels_gems_data
WHERE designer LIKE '%Van Cleef%'
  AND status = 'sold'
ORDER BY sale_date DESC
LIMIT 50;
```

DISCUSS

Questions for the room

- Why do we filter on status equals sold before computing pricing power, and what would change if unsold lots were included without adjustment?
- What does a pricing power ratio above 1.0 tell you about a brand, and can you think of a business reason a brand might intentionally set conservative estimates?
- Why is quarterly bucketing a reasonable choice for a demand index in this dataset, as opposed to monthly or annual bucketing?
- What are the tradeoffs of using AVG versus a percentile-based measure like median when summarizing realized prices?
- If two brands have the same average realized price but very different lot counts, how would you decide which one has stronger evidence behind its number?

HOMEWORK

This week's assignment

- Each student selects two brands from the jewelry or watches data and builds a quarterly demand index and a pricing power ratio for each, using only the sandbox SQL editor.
- The deliverable is a short technical writeup (three pages maximum) containing all four SQL queries used, the exported result tables or charts, and a written comparison of the two brands' demand index and pricing power.
- The writeup should explicitly note whether sold-lot counts are large enough in each quarter to trust the average, and should not extrapolate a growth trend from the last one or two quarters, since recent-period ingestion volume is not stable enough for period-over-period comparisons.
- Grading criteria:
- correctness of SQL (30 percent), correct construction of the demand index and pricing power ratio (25 percent), appropriate caveats about sample size and recency (20 percent), and clarity of the written comparison (25 percent).

NEXT

Wrapping up

- Restate the one-sentence takeaway before students leave.
- Point to the facilitator guide for discussion guidance and teaching notes.
- Questions or a class API key: info@altfndata.com