ALT/FNDATA ACADEMY · SQL AND FIELD CHEAT SHEET

SQL and field reference

A one-stop reference for the no-code sandbox at sandbox.altfndata.com. Every category table shares the same documented field vocabulary, so a query written for one category usually works on another by swapping the table name. No API key is needed for anything on this sheet.

The tables

The sandbox exposes one table per category. The most-used are below; the Coverage panel lists the rest, from wine and whisky to automobiles to books.

Table	Category
all_watches_data	Watches
all_jewels_gems_data	Jewelry and gems
all_handbags_data	Handbags
all_fine_art_data	Fine art
all_wine_whisky_data	Wine and whisky
all_automobile_data	Automobiles
all_books_data	Books

Documented fields

These are the fields you query. They are shared across the category tables.

Field	What it holds
designer	The brand or maker, such as Rolex or Hermes. Your main brand filter.
model	The model line within a brand, such as Daytona.
item_title	The full lot title or description. Use it for text search.
sale_date	The actual date the lot sold or was offered. Use this for anything time-based.
usd_price_decimal	The realized price in US dollars. Populated only when status is sold.
sale_estimates_high_usd_price	The pre-sale high estimate. A forecast, not what was paid.
status	sold or unsold. Filter to sold before you read a price.
vendor	The auction house or marketplace that listed the lot.
stock_ticker	The public-company ticker for the brand's parent, when one exists.

The three core metrics

- **Pricing power:** the median of usd_price_decimal divided by sale_estimates_high_usd_price, over sold lots with a positive estimate. Above 1.0 means lots clear above their high estimate.
- **Sell-through rate:** sold lots divided by all offered lots for a brand. A demand signal that does not depend on price.
- **Demand over time:** lot count and median price by quarter, using DATE_TRUNC on sale_date.

Query patterns

Peek at a table before you do anything else:

```
SELECT designer, model, item_title, sale_date, usd_price_decimal, status
FROM all_watches_data
LIMIT 5;
```

Sold lots for one brand, most recent first:

```
SELECT designer, model, item_title, sale_date, usd_price_decimal, vendor
FROM all_watches_data
WHERE designer = 'Rolex'
  AND status = 'sold'
ORDER BY sale_date DESC
LIMIT 20;
```

Filter by brand and a date range (use DATE literals):

```
SELECT designer, model, sale_date, usd_price_decimal, status
FROM all_watches_data
WHERE designer = 'Rolex'
  AND sale_date >= DATE '2024-01-01'
  AND sale_date <  DATE '2025-01-01'
ORDER BY sale_date;
```

Pricing power for a brand (median realized over high estimate):

```
SELECT designer,
  approx_percentile(usd_price_decimal / sale_estimates_high_usd_price, 0.5) AS pricing_power_median
FROM all_watches_data
WHERE status = 'sold'
  AND sale_estimates_high_usd_price > 0
  AND designer = 'Van Cleef & Arpels'
GROUP BY designer;
```

Sell-through rate for a brand:

```
SELECT designer,
  COUNT(*) FILTER (WHERE status = 'sold') AS sold_count,
  COUNT(*) AS offered_count,
  CAST(COUNT(*) FILTER (WHERE status = 'sold') AS DOUBLE) / COUNT(*) AS sell_through_rate
FROM all_watches_data
WHERE designer = 'Patek Philippe'
GROUP BY designer;
```

Demand over time, by quarter:

```
SELECT designer,
  DATE_TRUNC('quarter', sale_date) AS sale_quarter,
  COUNT(*) AS lots_sold,
  approx_percentile(usd_price_decimal, 0.5) AS median_price_usd
FROM all_watches_data
WHERE designer = 'Omega'
  AND status = 'sold'
GROUP BY designer, DATE_TRUNC('quarter', sale_date)
ORDER BY sale_quarter;
```

Link a brand to a public company through its ticker:

```
SELECT designer, item_title, sale_date, usd_price_decimal, stock_ticker
FROM all_watches_data
WHERE stock_ticker = 'CFR.SW'
  AND status = 'sold'
ORDER BY sale_date DESC
LIMIT 50;
```

Text search across lot titles (lower-case both sides):

```
SELECT designer, model, item_title, sale_date, usd_price_decimal
FROM all_watches_data
WHERE LOWER(item_title) LIKE '%chronograph%'
ORDER BY sale_date DESC
LIMIT 30;
```

An export-ready leaderboard (group, then keep only well-covered brands):

```
SELECT designer,
  approx_percentile(usd_price_decimal / sale_estimates_high_usd_price, 0.5) AS pricing_power_median,
  COUNT(*) AS sold_lot_count
FROM all_watches_data
WHERE status = 'sold'
  AND sale_estimates_high_usd_price > 0
GROUP BY designer
HAVING COUNT(*) >= 20
ORDER BY pricing_power_median DESC;
```

Habits that keep your answer honest

1. Filter status = 'sold' before you read usd_price_decimal. Unsold lots have no realized price.
2. Guard the pricing-power ratio with sale_estimates_high_usd_price > 0 so you never divide by zero or a blank estimate.
3. Use sale_date, never the date a row was added, for anything time-based. That is what point-in-time correctness means.
4. Treat the newest one or two quarters with care. A dip in recent volume is often recent sales not yet ingested, not a real fall in demand. Read counts alongside medians.
5. Check how many vendors a result rests on with GROUP BY vendor before you generalize. A brand's top prices can come from one or two houses.
6. Swap the table name to move a query to another category. The documented fields are shared, so the same pattern travels.

Full lessons and worked examples are in the tutorials and course modules at academy.altfndata.com. Questions or a class API key: info@altfndata.com.