ALT/FNDATA ACADEMY · PROBLEM SET

Problem set 16: Data science for art market research

Module 16. Total: 100 points. Format: no-code sandbox, with an optional API extension in Problem 5 using a shared class API key from your instructor.

Before you start

This problem set works with the fine art data, all_fine_art_data, using documented fields only: designer, item_title, sale_date, usd_price_decimal, sale_estimates_high_usd_price, status, and vendor. In this table the designer field holds the artist or maker name and the vendor field holds the auction house. Choose one well-known artist to use throughout Problems 1 through 3, and confirm the exact spelling in the designer field, along with roughly how many sold lots that artist has, using the coverage browser before you write any filter. For Problem 4 choose at least three artists to compare as a cohort. Confirm every field name you use in the schema browser, or via the GET /v1/tables documentation, since a misspelled or misremembered field name can silently return zero rows rather than an error. Remember throughout that item_title is free text, not a documented characteristic field, so any keyword drawn from it is an imperfect proxy, and that the data captures the secondary auction market only.

Problem 1 (25 points): Build the market index and read the recency caveat

Write a SQL query against the fine art data that builds your chosen artist's market index: the median of usd_price_decimal by year, restricted to sold records, with the count of sold lots shown alongside every period.

```
-- your query here
```

Then, in three to four sentences, identify which of the artist's own most recent one or two periods your query returns should be treated as provisional rather than as the current trend, explain why using what the count column shows for those periods, and state why a median index like this one measures method and the changing composition of what sold, not the pure appreciation of the artist's work.

Problem 2 (20 points): Demand signals that do not depend on price

Write SQL that computes, for your chosen artist, both sell-through rate (the share of offered lots that sold, with both sold and unsold in the denominator) and bought-in rate (the complement of sell-through). You may use one query or two; state which figure each query produces.

```
-- your query (or queries) here
```

Problem 3 (15 points): A rough hedonic-style split

Write a SQL query that splits your chosen artist's sold lots into medium buckets using a keyword drawn from item_title (for example print, photograph, painting, and an other-or-unspecified bucket for anything that does not match), and reports the median of usd_price_decimal and the count of sold lots for each bucket.

```
-- your query here
```

In two to three sentences, name one kind of lot this keyword approach would misclassify, and state one caveat you would attach to any conclusion drawn from this split.

Problem 4 (20 points): Cohort comparison across artists

Write a SQL query that compares your chosen cohort of at least three artists at once: group by designer, report the count of sold lots, the count of offered lots, the sell-through rate, and the median of usd_price_decimal, restrict the table to artists with at least a minimum number of lots using a HAVING clause, and order the result by median realized value. State the minimum-lot threshold you chose and why.

```
-- your query here
```

In two to three sentences, explain why the minimum-lot threshold is necessary before an artist appears in a comparative table, and state the secondary-market-only caveat that applies to every column in this table.

Problem 5 (20 points): One artist via the API, index built client-side (optional code extension)

Using the shared class API key your instructor has provided, write the endpoint and JSON request body you would send to POST /v1/tables/all_fine_art_data/query to retrieve the raw, documented fields needed to build your chosen artist's median-by-year index (designer, item_title, sale_date, usd_price_decimal), filtered to that artist and to sold records.

```
POST /v1/tables/<table_name>/query
Header: X-API-Key: <class key>
Body:
{
  ...
}
```

Then, in three to four sentences, explain why the query endpoint returns rows rather than a server-side median, how you would paginate on offset to retrieve the artist's full sold-lot history, and how you would group the rows by year and compute the median in pandas once every page is collected, so that the result reproduces what the SQL editor computed server-side in Problem 1.

Submission. Turn in this file with your SQL, JSON body, and written answers filled in.