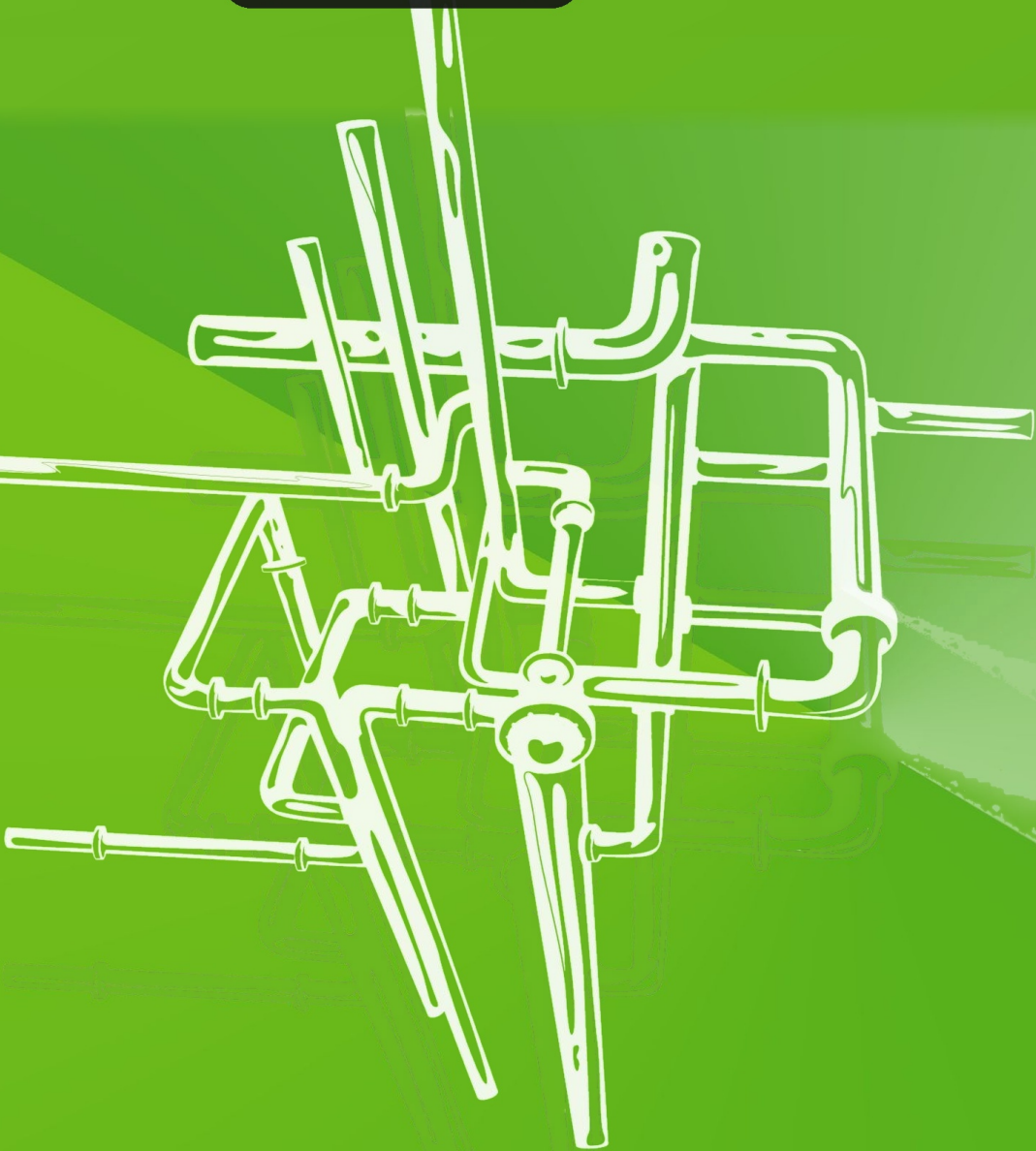


MongoDB Aggregation Pattern Pack

Vol. 1: Reporting & Dashboards

Python / PyMongo



John Lynn

MongoDB Aggregation Pattern Pack, Vol. 1: Reporting & Dashboards (Python)

Python / PyMongo · Copy-paste reporting patterns
against Atlas sample data

John Lynn

This book is available at <https://leanpub.com/mongo-agg-patterns-python>

This version was published on 2026-10-12



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2026 John Lynn

Contents

About	1
How to read these patterns	3
Leading commas in Python	3
pymongo notes	3
Sample data	5
On Atlas	5
On local MongoDB	6
What a sale looks like	7
How to run the examples	8
PyMongo setup	8
A smoke test	8
Running a pattern	9
Compass and mongosh	9
Pattern 1 – Time-bucket totals	11
When this pattern fits	11
When it does not fit	11
Sample document (trimmed from Atlas)	11
Pipeline	12
Expected output shape	14
Watch out	15
Variations	15
Pattern 2 – Top-N per category	16
When this pattern fits	16
When it does not fit	16
Sample data	16
Pipeline	16

CONTENTS

Expected output shape	16
Watch out	16
Variations	17
Pattern 3 – Running totals / cumulative series	18
When this pattern fits	18
When it does not fit	18
Sample data	18
Pipeline	18
Expected output shape	18
Watch out	19
Variations	19
Pattern 4 – Period-over-period	20
When this pattern fits	20
When it does not fit	20
Sample data	20
Pipeline	20
Expected output shape	20
Watch out	20
Variations	21
Pattern 5 – Null-safe metrics	22
When this pattern fits	22
When it does not fit	22
Sample data	22
Pipeline	22
Expected output shape	22
Watch out	22
Variations	23
Pattern 6 – Multi-collection report via \$lookup	24
When this pattern fits	24
When it does not fit	24
Sample data	24
Pipeline	24
Expected output shape	24
Watch out	24
Variations	25

Pattern 7 – Faceted dashboard payload	26
When this pattern fits	26
When it does not fit	26
Sample data	26
Pipeline	26
Expected output shape	26
Watch out	26
Variations	27
Pattern 8 – Histogram / distribution	28
When this pattern fits	28
When it does not fit	28
Sample data	28
Pipeline	28
Expected output shape	28
Watch out	28
Variations	29
Pattern 9 – Active vs churned in a window	30
When this pattern fits	30
When it does not fit	30
Sample data	30
Pipeline	30
Expected output shape	30
Watch out	30
Variations	31
Pattern 10 – Pagination-friendly report pages	32
When this pattern fits	32
When it does not fit	32
Sample data	32
Pipeline – page and count	32
Keyset variation (preferred for deep pages)	32
Expected output shape (facet form)	32
Watch out	33
Variations	33
What's next	34

About

Welcome! If you need reporting pipelines you can run and adapt, this is a good place to start.

If you've ever stared at an empty aggregation pipeline with a deadline close, you know why this book exists. Maybe you needed sales by week, top products, or this month against last month. Maybe the dashboard needed all of it in one round trip. The MongoDB docs are great, but you also need a few tested reporting patterns that you can run as they are and change to fit your data.

That's what this is: a pattern pack, not a catalog of every aggregation operator. Ten practical recipes for everyday reporting and dashboard work. Each chapter starts from a real question, shows a pipeline you can run against Atlas sample data (we lean on `sample_supplies` a lot), and talks through when the pattern fits, when it doesn't, and the common mistakes to watch for.

What I like to do is keep the pipelines honest and editable. You'll see leading commas, even in Python lists, so stages are easy to comment out while you test. You'll see the same habits I use when I'm building something for a real report, not a slide deck. The goal is a clear answer you can ship, not clever aggregation.

If you know basic `$match` and `$group` and can run a Python script against MongoDB, you're ready. If aggregation is new to you, skim the MongoDB docs first. Then come back when you want working examples to help apply what you learned. Plenty of good data people start right here.

Volume 1 covers reporting and dashboards. The examples use Python and PyMongo. The same ten patterns also come in a Node and mongosh edition, so you can work in the language you like best. The aggregation stages are identical in both books; only the client code differs. If this book helps you, I'd love to write more. Later volumes could cover more reporting, ops diagnostics, or domain packs, all in the same hands-on format.

Examples target MongoDB 7.x / 8.x unless a chapter notes an earlier requirement (for example `$setWindowFields` on 5.0+), and PyMongo 4.x or later. Version-sensitive behavior is called out where it matters. Corrections and clarifications are always welcome; Leanpub makes it easy to keep a living manuscript as the material improves.

You're in the right place. Open **How to read**, load the sample data, and start with Pattern 1. Then jump to the chapter that matches the report you're building.

Happy building – and thank you for reading!

How to read these patterns

Each pattern follows a consistent structure:

- Problem statement
- When it fits / when it does not
- Sample collection
- Pipeline (leading-comma style Python lists)
- Expected output shape
- Watch outs
- Short variations

Leading commas in Python

Aggregation examples in this pack use a leading comma on every stage after the first. That style is valid in Python lists:

```
1 pipeline = [  
2     { "$match": { # ... } }  
3     , { "$group": { # ... } }  
4     , { "$sort": { # ... } }  
5 ]
```

The first stage has no comma. Later stages begin with a comma. That layout makes it easier to reorder, insert, delete, or comment out stages without adjusting a trailing comma on the previous line.

While developing a pipeline, comment out later stages and advance one stage at a time. Starting from `$match` (often with `$limit`) is a practical way to inspect intermediate results.

pymongo notes

1. Prefer Atlas sample data so examples are reproducible.
2. Apply `$match` early when it reduces work for later stages.
3. Use the same timezone for date truncation that you use for display.
4. Pass native `datetime` objects (timezone-aware UTC is clearest for bounds) into `$match`; pymongo encodes them as BSON dates.
5. Money fields from Atlas sample data often arrive as `bson.Decimal128`. Convert explicitly when you need floats for charts; keep `Decimal128` when you need exact decimal arithmetic.
6. Call `list(collection.aggregate(pipeline))` or iterate the cursor; do not assume a single document return.

Sample data

The pipelines in this volume use the official MongoDB Atlas sample datasets. You need:

- **sample_supplies.sales** – Patterns 1–5 and 7–10
- **sample_mflix.movies** and **sample_mflix.comments** – Pattern 6 (\$lookup)

Load that data into **Atlas** or into a **local MongoDB**. The collection names and document shapes are the same either way.

Everything you need to load the samples is in this chapter: Atlas steps, local restore, Windows notes, and the Pattern 6 index. If a chapter needs a join, it says so up front.

* * *

On Atlas

1. Open your cluster in [MongoDB Atlas](#).
2. On the cluster card, open the  menu and choose **Load Sample Dataset**.
3. Wait until the load finishes, then connect with mongosh, Compass, or PyMongo.

```

1 from pymongo import MongoClient
2
3 client = MongoClient("<your Atlas URI>") # or mongodb://127.0.0.1:27017 for
  ↳ local
4 print(client["sample_supplies"]["sales"].count_documents({}))
5 print(client["sample_supplies"]["sales"].find_one())
6 print(client["sample_mflix"]["movies"].count_documents({}))
7 print(client["sample_mflix"]["comments"].count_documents({}))

```

Expect about **5,000** sales, **about 21,000** movies, and **about 41,000** comments.

An M0 free cluster is enough for most chapter examples. Heavier aggregations (especially Pattern 6) can hit free-tier limits. If that happens, switch to local MongoDB. Pattern 6 also benefits from an index on `comments.movie_id`:

```

1 client["sample_mflix"]["comments"].create_index("movie_id")

```

* * *

On local MongoDB

1. Install and start [MongoDB Community Server](#). The default port is **27017**.
2. Download and restore the official archive:

```

1 curl -L https://atlas-education.s3.amazonaws.com/sampleddata.archive -o
  ↳ sampleddata.archive
2 mongorestore --archive=sampleddata.archive --host=127.0.0.1 --port=27017

```

On Windows PowerShell, use `curl.exe` and `mongorestore --archive=sampleddata.archive --port=27017`. Install [Database Tools](#) if `mongorestore` is missing.

3. Verify with the same count checks as above, pointing `MongoClient` at `mongodb://127.0.0.1:27017`.

Prefer local MongoDB when you want to rerun pipelines quickly without Atlas free-tier limits.

* * *

What a sale looks like

Each `sample_supplies.sales` document is one sale: `saleDate`, `items` (with price and quantity), `storeLocation`, `customer`, `purchaseMethod`, and `couponUsed`.

Revenue comes from line items. Use `$unwind` or `$reduce` over `items` when totaling money. Item price values are typically `Decimal128` in the sample dataset.

Secondary collection (when a chapter needs a join): `sample_mflix`. Chapters that require it say so explicitly.

How to run the examples

Once the sample data is loaded (see **Sample data**), you can run any chapter pipeline the same way.

PyMongo setup

1. Use Python 3.10 or later. A virtual environment keeps the driver separate from other projects.
2. Install the driver:

```
1 python3 -m venv .venv
2 source .venv/bin/activate
3 pip install pymongo
```

On Windows PowerShell, activate the environment with `.venv\Scripts\Activate.ps1` instead of `source`.

3. Set `MONGODB_URI` to your Atlas connection string, or to `mongodb://127.0.0.1:27017` for local MongoDB:

```
1 export MONGODB_URI="mongodb://127.0.0.1:27017"
```

On Windows PowerShell, use `$env:MONGODB_URI = "mongodb://127.0.0.1:27017"`.

A smoke test

Before you run a full pattern, check that the connection and the data are in place. Save this as `smoke_test.py`:

```
1 import os
2
3 from pymongo import MongoClient
4
5 client = MongoClient(os.environ.get("MONGODB_URI",
  ↳ "mongodb://127.0.0.1:27017"))
6 sales = client["sample_supplies"]["sales"]
7
8 pipeline = [
9     { "$limit": 3 }
10    # paste later stages from a pattern chapter here
11 ]
12
13 for doc in sales.aggregate(pipeline):
14     print(doc)
```

Run it:

```
1 python smoke_test.py
```

You should see three sale documents. You can also start python and paste the same lines into the interactive prompt, which is convenient when you want to inspect results one stage at a time.

Running a pattern

1. Open the chapter and use the database it names (usually `sample_supplies`; Pattern 6 uses `sample_mflix`).
2. Copy the chapter's `pipeline = [...]` block over the smoke-test pipeline, together with any parameters it defines (date bounds, limits, sort keys). Set those first, then run.
3. Iterate the cursor, or call `list(collection.aggregate(pipeline))` when the result is small.

The chapter examples write the connection string inline, for example `MongoClient("mongodb://127.0.0.1:27017")`, so each listing stands on its own. Replace it with your Atlas URI, or with `os.environ["MONGODB_URI"]`, when you run against another server.

Keep the leading-comma style from **How to read these patterns** so you can comment stages in and out while you learn.

Compass and mongosh

In MongoDB Compass, open the same database and collection, go to the **Aggregations** tab, and rebuild the stages from the chapter (or paste them if your Compass version accepts a full pipeline). The results should match PyMongo for the same data.

The stages are the same in mongosh. Only the client syntax differs: mongosh does not require quoted keys, and it uses `ISODate(...)` and `NumberDecimal(...)` where Python uses `datetime` and `Decimal128`. PyMongo is enough to follow along with this book.

The Python tests for these patterns are optional and live at <https://github.com/johnlyn763/mongo-agg-patterns/tree/main/python>. You do not need them to follow the book.

If you would rather copy a pipeline without line numbers, every pattern is in the repo as plain Python at <https://github.com/johnlyn763/mongo-agg-patterns/tree/main/python/patterns>.

Pattern 1 – Time-bucket totals

This pattern produces revenue and counts by day, week, or month for a time-series chart.

A common mistake is to format dates as strings and `$group` on those strings, or to truncate timestamps in UTC while the chart axis uses another timezone. Day boundaries then disagree with what the user sees, especially around midnight and around daylight-saving changes.

The examples use the Atlas sample dataset so you can run them without preparing your own data. Load **Sample Dataset** in Atlas, then use the `sample_supplies.sales` collection. Each document is one sale. Line-item amounts are in the `items` array (price and quantity). Sale-level revenue is the sum of `price × quantity` across those items.

When this pattern fits

- Charts with a continuous time axis
- Daily, weekly, or monthly totals from `saleDate`
- Bucket sizes that `$dateTrunc` supports directly (day, week, month, and related units)

When it does not fit

- Fiscal calendars or custom business weeks – use a calendar collection or precomputed period keys
- Fine-grained histograms (for example sub-second or arbitrary numeric ranges) – use `$bucket` or `$bucketAuto`
- Charts that must show periods with no activity as zeros – this pipeline returns only buckets that contain data; fill missing periods in the application, or `$unionWith` a calendar of dates

Sample document (trimmed from Atlas)

```

1 {
2   "_id": ObjectId("5bd761dcae323e45a93ccfe8"),
3   "saleDate": datetime(2015, 3, 23, 21, 6, 49, 506000), # UTC in BSON
4   "items": [
5     { "name": "notepad", "price": Decimal128("35.29"), "quantity": 2 },
6     { "name": "pens", "price": Decimal128("56.12"), "quantity": 5 },
7     # ...
8   ],
9   "storeLocation": "Denver",
10  "purchaseMethod": "Online",
11  "couponUsed": True,
12  "customer": {
13    "gender": "M",
14    "age": 42,
15    "email": "cauho@witwuta.sv",
16    "satisfaction": 4,
17  },
18 }

```

In pymongo, BSON dates decode to timezone-aware or naive datetime depending on codec options; Decimal money fields decode to bson.Decimal128.

Pipeline

Pipelines in this pack use **leading commas**: the first stage has no comma; each later stage begins with a comma. That makes it easier to reorder, insert, delete, or comment out stages without fixing a trailing comma on the previous line. The same layout works in Python lists.

```

1 from datetime import datetime, timezone
2
3 from bson.decimal128 import Decimal128
4 from pymongo import MongoClient
5
6 client = MongoClient("mongodb://127.0.0.1:27017") # or your Atlas URI
7 sales = client["sample_supplies"]["sales"]
8
9 timezone_name = "America/New_York" # same timezone the chart displays
10 unit = "day"                       # "day" | "week" | "month"
11
12 start = datetime(2015, 1, 1, tzinfo=timezone.utc)
13 end = datetime(2016, 1, 1, tzinfo=timezone.utc)
14
15 pipeline = [
16   # Restrict the range early. In production, an index on saleDate is usually
    ↪ appropriate.

```

```

17   { "$match": {
18       "saleDate": {
19           "$gte": start,
20           "$lt": end,
21       }
22   }
23 }
24 # One document per line item, so price × quantity is straightforward
25 , { "$unwind": "$items" }
26 , { "$set": {
27     "lineRevenue": { "$multiply": ["$items.price", "$items.quantity"] }
28 }
29 }
30 # Truncate to the bucket start in the display timezone, then aggregate
31 , { "$group": {
32     "_id": {
33         "dateTrunc": {
34             "date": "$saleDate",
35             "unit": "unit",
36             "timezone": timezone_name,
37             # optional for weeks: "startOfWeek": "monday"
38         }
39     },
40     "revenue": { "$sum": "$lineRevenue" },
41     "lineItems": { "$sum": 1 },
42     "sales": { "$addToSet": "$_id" }, # distinct sale ids in the bucket
43 }
44 }
45 , { "$project": {
46     "_id": 0,
47     "bucketStart": "$_id",
48     "revenue": 1,
49     "lineItems": 1,
50     "sales": { "$size": "$sales" },
51 }
52 }
53 , { "$sort": { "bucketStart": 1 } }
54 ]
55
56 results = list(sales.aggregate(pipeline))

```

When developing a pipeline, it is useful to comment out later stages and work forward from `$match` (often with `$limit`) so you can inspect intermediate results.

```

1 explore = [
2   { "$match": {
3     "saleDate": {
4       "$gte": start,
5       "$lt": end,
6     }
7   }
8 }
9 , { "$limit": 3 }
10 # , { "$unwind": "$items" }
11 # , { "$set": { "lineRevenue": { "$multiply": ["$items.price",
12   ↪ "$items.quantity"] } } }
13 ]
14
15 preview = list(sales.aggregate(explore))

```

Expected output shape

```

1 [
2   {
3     "bucketStart": datetime(...), # start of that local calendar day
4     "revenue": Decimal128(...),  # or convertible numeric
5     "lineItems": 12,
6     "sales": 4,
7   },
8   # periods with no matching sales are omitted
9 ]

```

Exact values depend on the sample data loaded in your cluster. The field names and types above are the contract for the chart.

Decimal128 in Python

\$sum over Decimal128 line revenues typically yields Decimal128. For charting libraries that expect floats:

```
1 def to_float(value):
2     if isinstance(value, Decimal128):
3         return float(value.to_decimal())
4     return float(value)
5
6 for row in results:
7     row["revenue_f"] = to_float(row["revenue"])
```

Prefer keeping `Decimal128` (or `decimal.Decimal`) when you need exact money totals; convert only at the presentation boundary.

Watch out

Timezone. If `$dateTrunc` uses UTC and the chart labels use `America/New_York`, bucket boundaries will not match the axis. Pass the same `timezone` into `$dateTrunc` that you use for display.

Revenue. Summing `items.price` without multiplying by `quantity`, or without `$unwind` / `$reduce` over `items`, understates revenue.

Distinct sales after \$unwind. `$addToSet: "$_id"` still identifies unique sales because `_id` is the sale id, but large buckets can use significant memory. If you only need a sale count, aggregate that before `$unwind`, or report line-item counts only.

Datetime bounds. Pass `datetime` objects into `$match`, not date strings. UTC-aware bounds (`tzinfo=timezone.utc`) avoid ambiguity when comparing to BSON dates stored in UTC.

Variations

- **By store:** include `storeLocation` in the `$group _id` together with the `$dateTrunc` expression.
- **Sale count only:** omit `$unwind` and `$set`; `$group` by `$dateTrunc` of `saleDate` with `"sales": { "$sum": 1 }`.

Apply `$match` as early as practical so later stages process fewer documents.

Pattern 2 – Top-N per category

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When this pattern fits

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When it does not fit

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Sample data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Expected output shape

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Watch out

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Variations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pattern 3 — Running totals / cumulative series

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When this pattern fits

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When it does not fit

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Sample data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Expected output shape

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Decimal128 in Python

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Watch out

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Variations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pattern 4 — Period-over-period

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When this pattern fits

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When it does not fit

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Sample data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Expected output shape

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Watch out

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Variations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pattern 5 – Null-safe metrics

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When this pattern fits

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When it does not fit

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Sample data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Expected output shape

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Watch out

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Variations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pattern 6 — Multi-collection report via \$lookup

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When this pattern fits

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When it does not fit

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Sample data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Expected output shape

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Watch out

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Variations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pattern 7 — Faceted dashboard payload

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When this pattern fits

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When it does not fit

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Sample data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Expected output shape

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Watch out

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Variations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pattern 8 – Histogram / distribution

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When this pattern fits

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When it does not fit

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Sample data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Expected output shape

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Watch out

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Variations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pattern 9 — Active vs churned in a window

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When this pattern fits

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When it does not fit

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Sample data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Expected output shape

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Watch out

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Variations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pattern 10 — Pagination-friendly report pages

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When this pattern fits

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

When it does not fit

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Sample data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Pipeline — page and count

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Keyset variation (preferred for deep pages)

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Expected output shape (facet form)

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Watch out

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

Variations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.

What's next

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/mongo-agg-patterns-python>.