Picture Database

CIS 386 Database Management Systems

Brendan Short

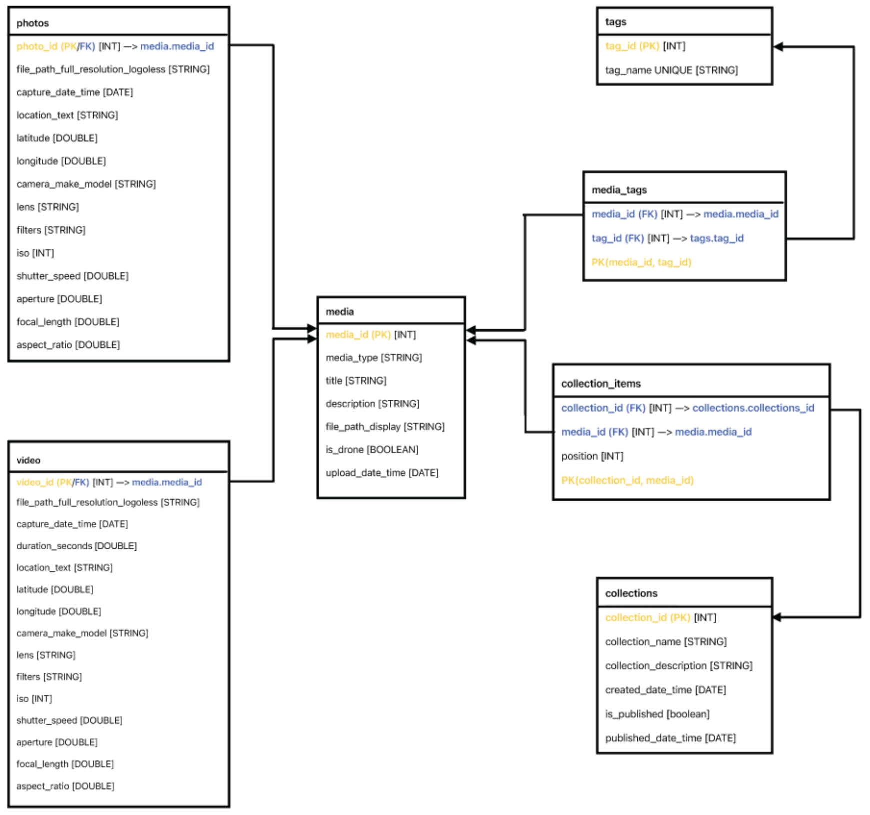
12/3/25

Organization Info and Requirements

- Client: my dad, hobby landscape & drone photographer in Northern Michigan.
- Current state: everything on a single external drive, loose folders, no real catalog/search.
- Risks & pain points: hard to find “that one photo,” single point of failure with everything stored on one drive.
- Requirements: searchable library, tagging, collections, video showcase.

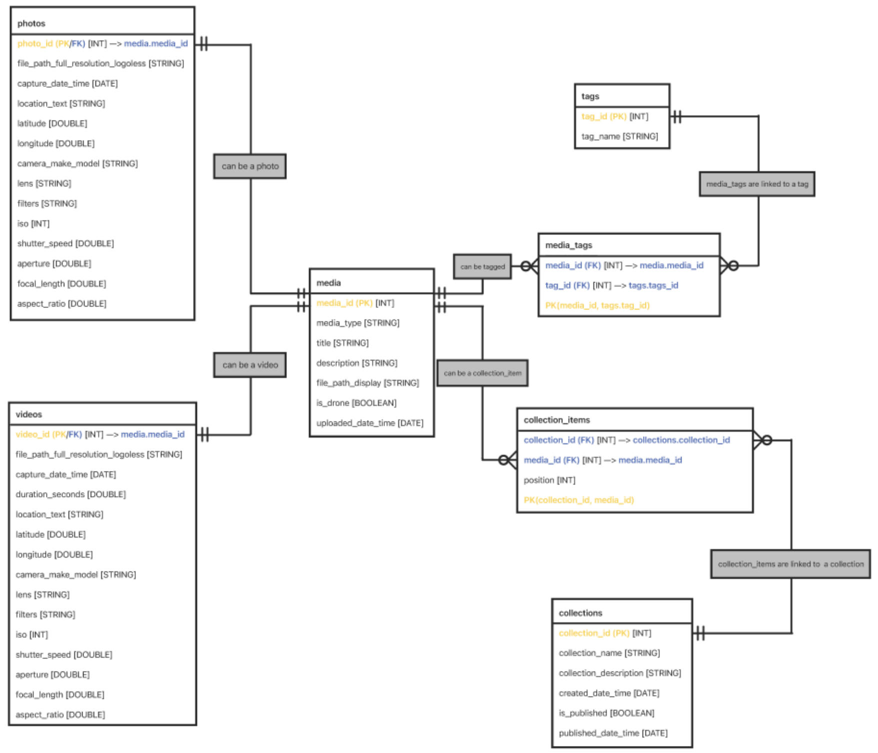
Schema

- Central **media** table: one row per photo or video (title, description, display file path, drone flag, upload datetime).
- Subtype tables **photos** and **videos** share the same PK as media for photo/video-specific EXIF fields.
- Tagging via **tags** + **media_tags** (zero-to-many).
- Curated collections of media via **collections** + **collection_items** (zero-to-many with position attribute so collections can be ordered whichever way they want).



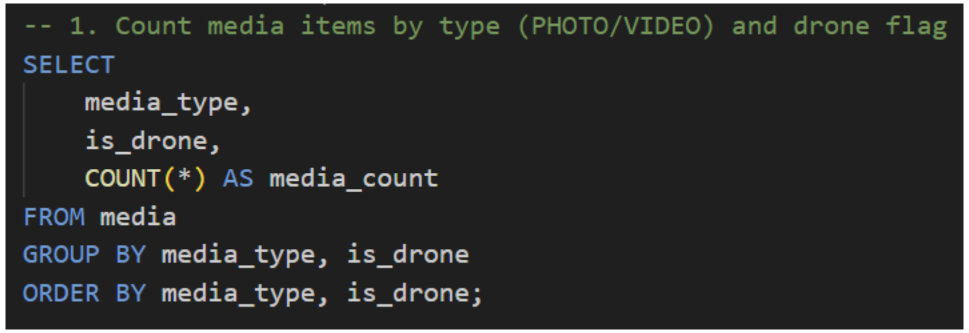
ER Diagram

- **media** at the center
- 1–1 from **media** → **photos** and **media** → videos
    - Media can be considered a supertype with photos and videos begin a subtype
- 0–M between **media** and **tags** via **media_tags** (association table)
- 0–M between **media** and **collections** via **collection_items** (association table).

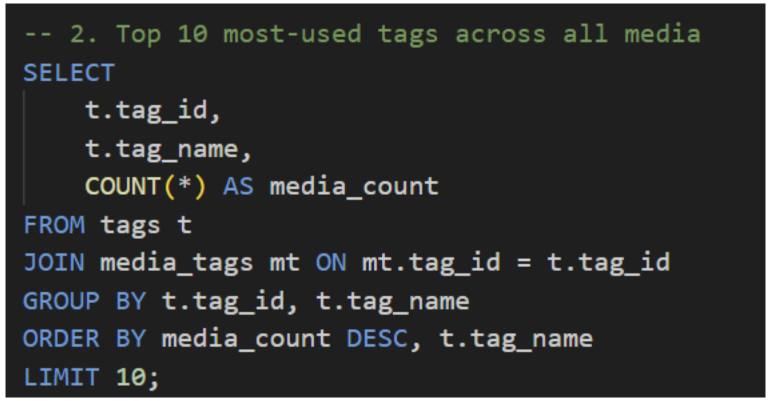


MySQL Data Analytics: 5 Questions

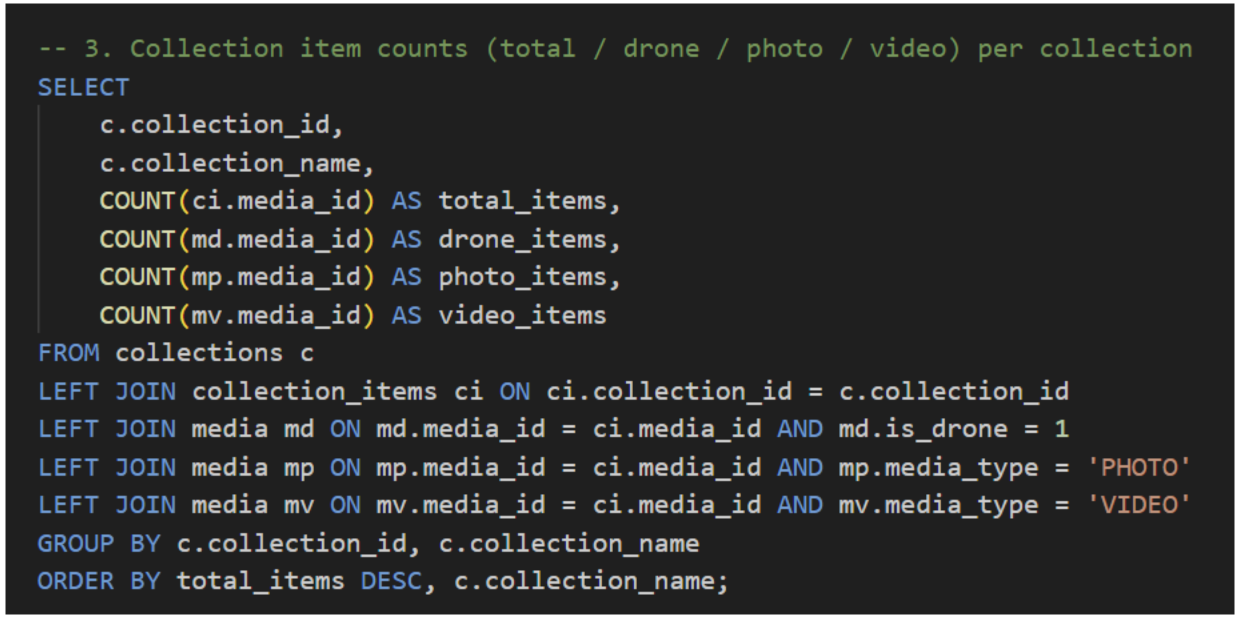
1. How many media items do we have by type (PHOTO vs VIDEO) and by drone vs non-drone?
2. What are the top 10 most-used tags across all media?
3. For each collection, how many total items, drone items, photos, and videos does it contain?
4. For 2025, how many photos were captured per month?
5. For each tag, what percentage of tagged media are drone media?



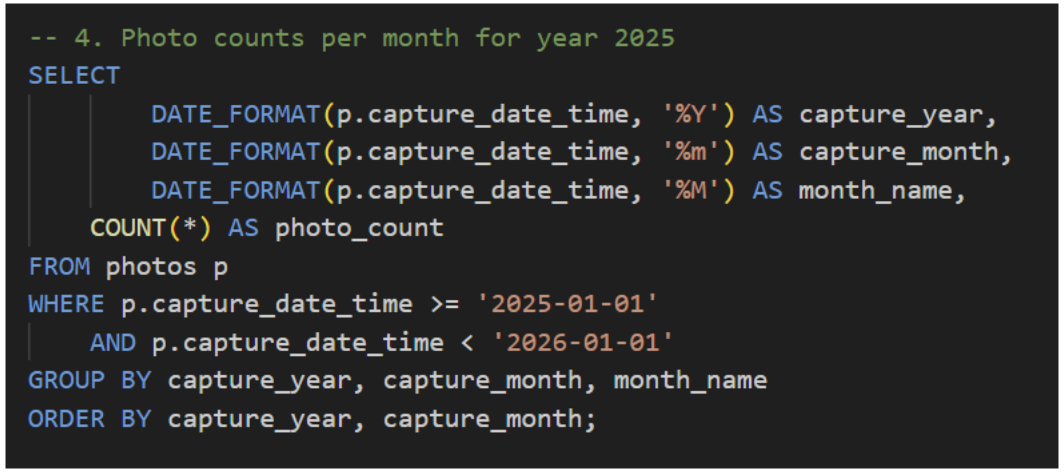
```sql
-- 1. Count media items by type (PHOTO/VIDEO) and drone flag
SELECT
    media_type,
    is_drone,
    COUNT(*) AS media_count
FROM media
GROUP BY media_type, is_drone
ORDER BY media_type, is_drone;
```



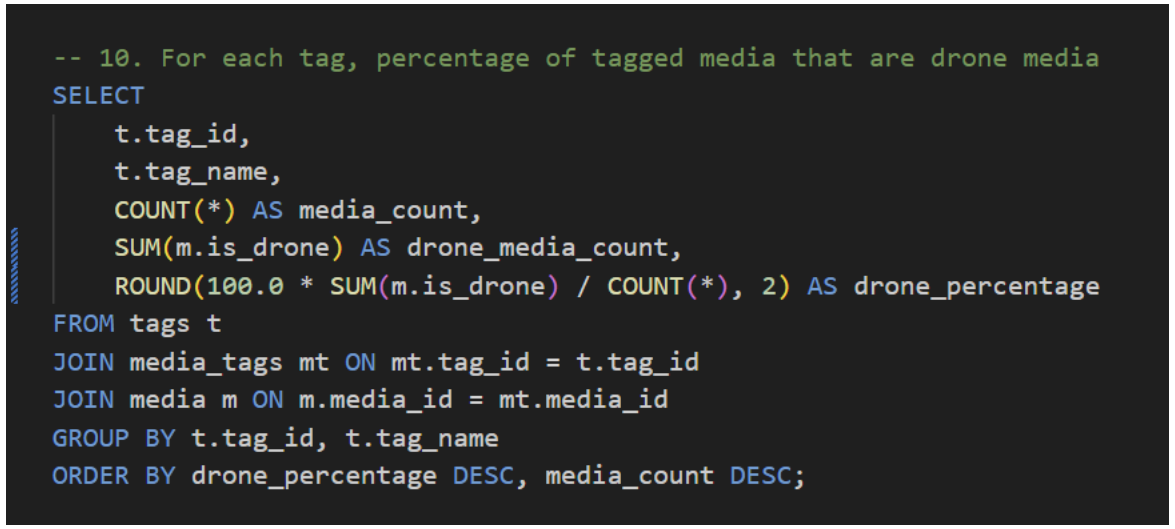
```sql
-- 2. Top 10 most-used tags across all media
SELECT
    t.tag_id,
    t.tag_name,
    COUNT(*) AS media_count
FROM tags t
JOIN media_tags mt ON mt.tag_id = t.tag_id
GROUP BY t.tag_id, t.tag_name
ORDER BY media_count DESC, t.tag_name
LIMIT 10;
```



```sql
-- 3. Collection item counts (total / drone / photo / video) per collection
SELECT
    c.collection_id,
    c.collection_name,
    COUNT(ci.media_id) AS total_items,
    COUNT(md.media_id) AS drone_items,
    COUNT(mp.media_id) AS photo_items,
    COUNT(mv.media_id) AS video_items
FROM collections c
LEFT JOIN collection_items ci ON ci.collection_id = c.collection_id
LEFT JOIN media md ON md.media_id = ci.media_id AND md.is_drone = 1
LEFT JOIN media mp ON mp.media_id = ci.media_id AND mp.media_type = 'PHOTO'
LEFT JOIN media mv ON mv.media_id = ci.media_id AND mv.media_type = 'VIDEO'
GROUP BY c.collection_id, c.collection_name
ORDER BY total_items DESC, c.collection_name;
```



```sql
-- 4. Photo counts per month for year 2025
SELECT
        DATE_FORMAT(p.capture_date_time, '%Y') AS capture_year,
        DATE_FORMAT(p.capture_date_time, '%m') AS capture_month,
        DATE_FORMAT(p.capture_date_time, '%M') AS month_name,
    COUNT(*) AS photo_count
FROM photos p
WHERE p.capture_date_time >= '2025-01-01'
    AND p.capture_date_time < '2026-01-01'
GROUP BY capture_year, capture_month, month_name
ORDER BY capture_year, capture_month;
```



```sql
-- 10. For each tag, percentage of tagged media that are drone media
SELECT
    t.tag_id,
    t.tag_name,
    COUNT(*) AS media_count,
    SUM(m.is_drone) AS drone_media_count,
    ROUND(100.0 * SUM(m.is_drone) / COUNT(*), 2) AS drone_percentage
FROM tags t
JOIN media_tags mt ON mt.tag_id = t.tag_id
JOIN media m ON m.media_id = mt.media_id
GROUP BY t.tag_id, t.tag_name
ORDER BY drone_percentage DESC, media_count DESC;
```

Problems With MySQL

- A lot of effort: multi-step manual ingestion (Lightroom export → ExifTool → CSV cleanup → SQL INSERT scripts.
- Rigid schema: adding new EXIF fields means schema changes and new ingestion queries.
- Not scalable: current pipeline doesn’t scale to hundreds or thousands of new files per month.

Solution With MongoDB

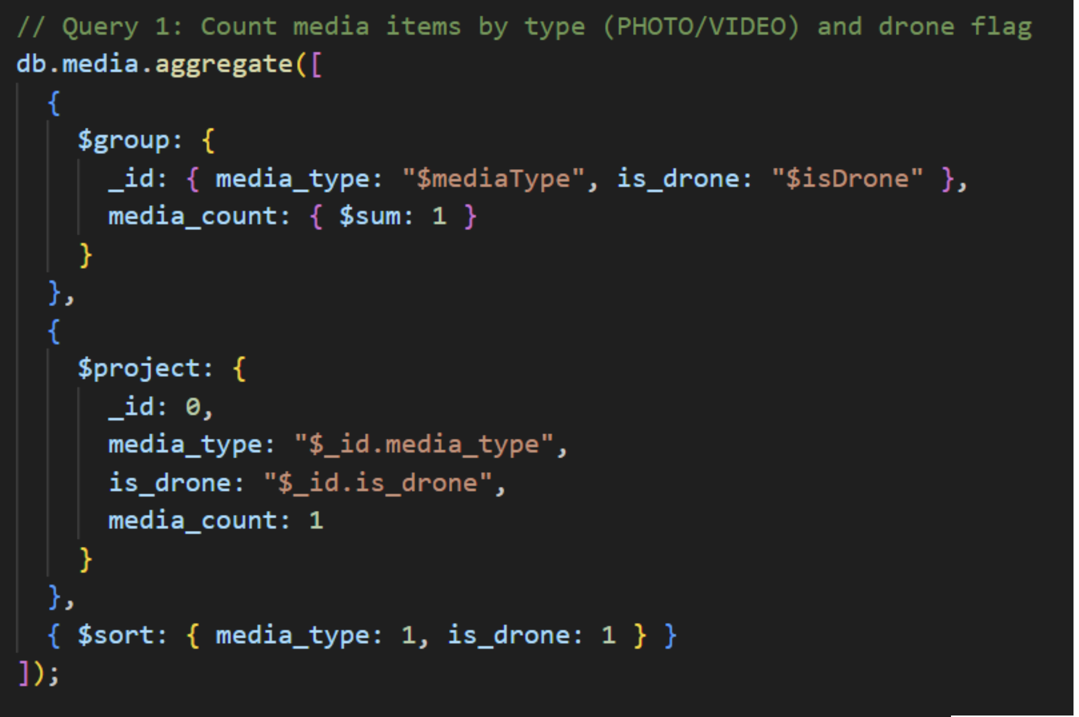
- One document per media item, instead of spreading data across 5+ tables.
- Embed EXIF/capture metadata, tags, and collection membership inside each document.
- Handle missing or extra fields naturally (schema flexibility).
- Use aggregation pipelines to answer the same analysis questions as MySQL.

MongoDB Schema

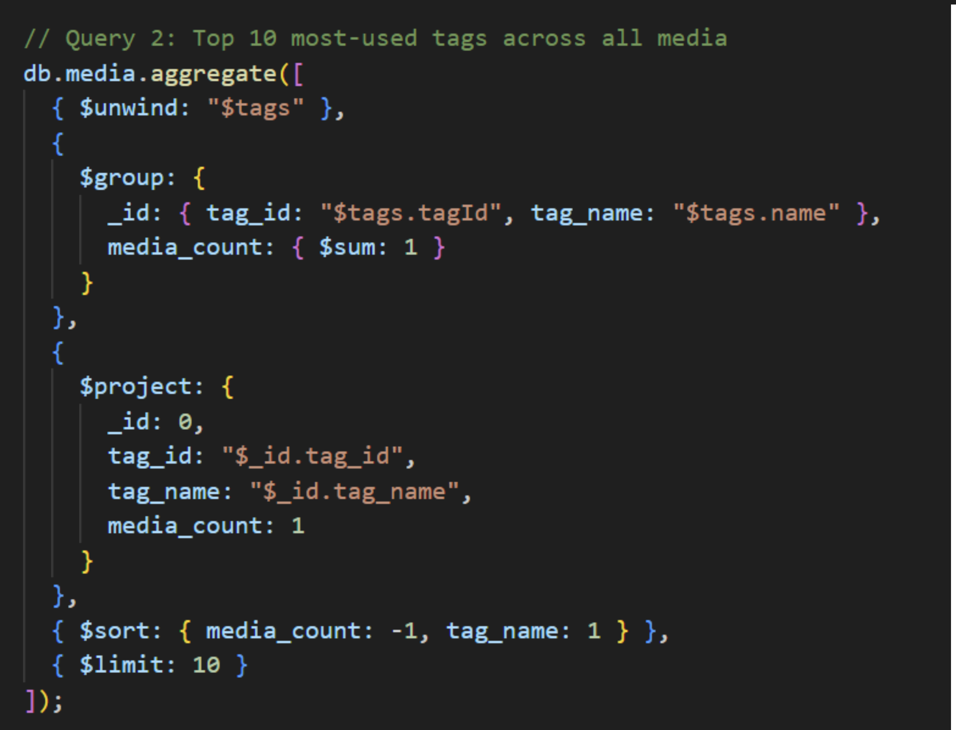
- Single media collection; each document in this collection is a photo or video.
- Embedded fields and subdocuments to construct the media object:
  - capture: dateTime, durationSeconds
  - camera: makeModel, iso, shutterSpeed, aperture, focalLength, etc.
  - tags: array of { tagId, name }
  - collections: array of { collectionId, name }
- Everything fits into one collection, which simplifies reads for a single media item.
- Tradeoff: aggregation queries are more complex (lots of $unwind, conditional logic) compared to SQL joins.

Data Analytics With MongoDB: 5 Questions

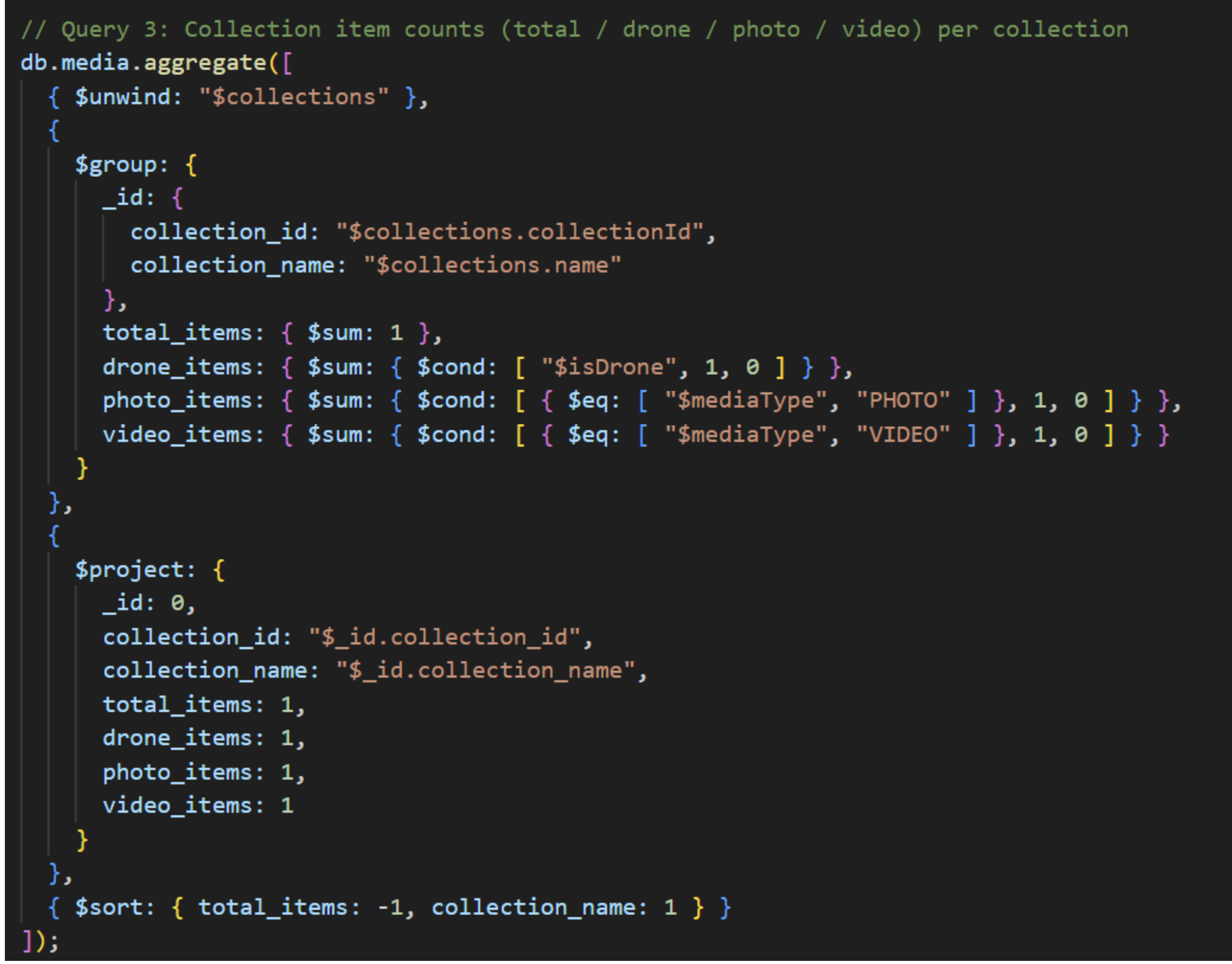
1. Count media items by type and drone flag using $group on mediaType and isDrone.
2. Find the top 10 tags by counting how often each tag appears in tags (using $unwind).
3. For each collection, compute total items and counts of drone, photo, and video items.
4. For 2025 photos, group by capture month and count items.
5. For each tag, compute what percentage of tagged media are drone media.



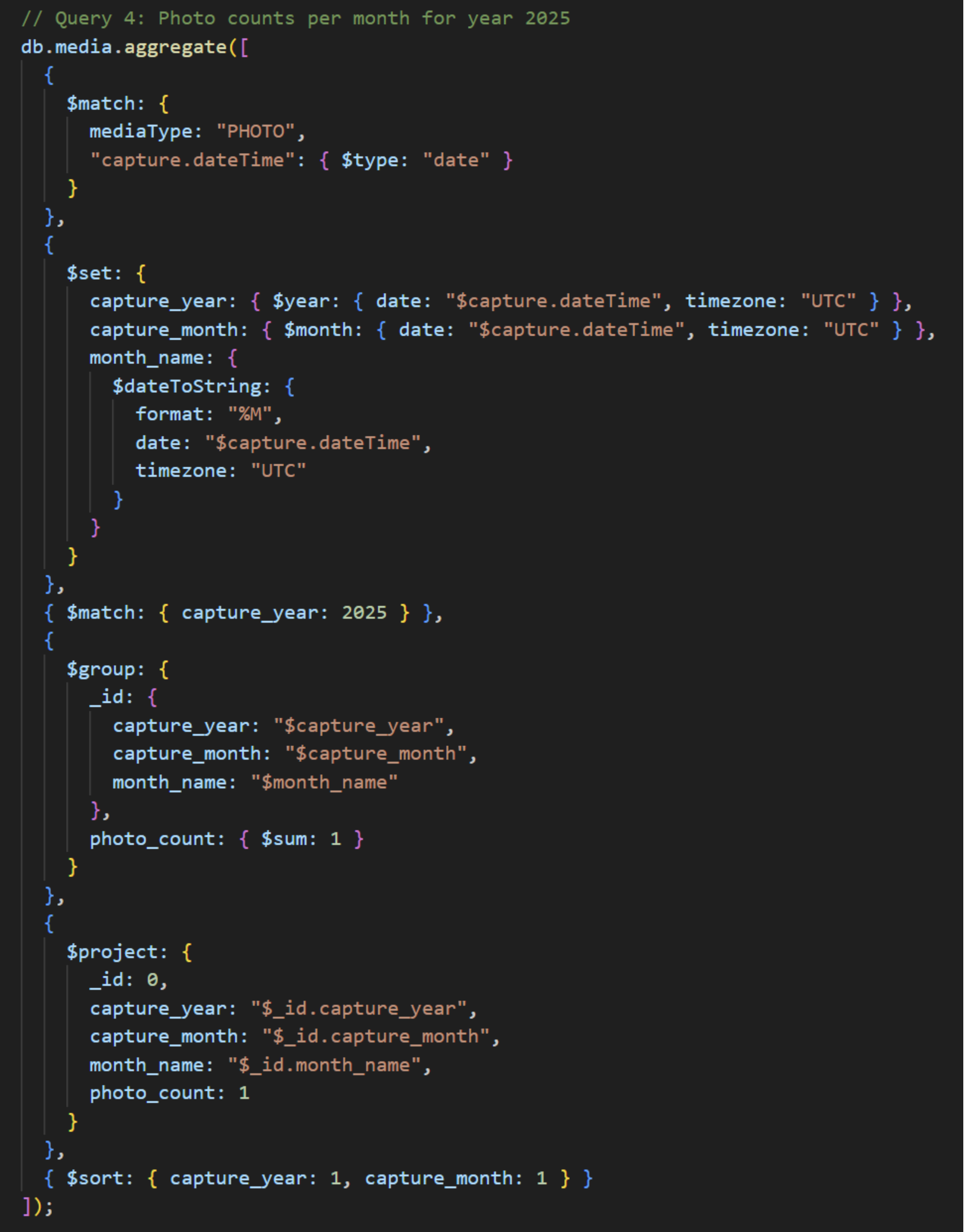
```
// Query 1: Count media items by type (PHOTO/VIDEO) and drone flag
db.media.aggregate([
  {
    $group: {
      _id: { media_type: "$mediaType", is_drone: "$isDrone" },
      media_count: { $sum: 1 }
    }
  },
  {
    $project: {
      _id: 0,
      media_type: "$_id.media_type",
      is_drone: "$_id.is_drone",
      media_count: 1
    }
  },
  { $sort: { media_type: 1, is_drone: 1 } }
]);
```



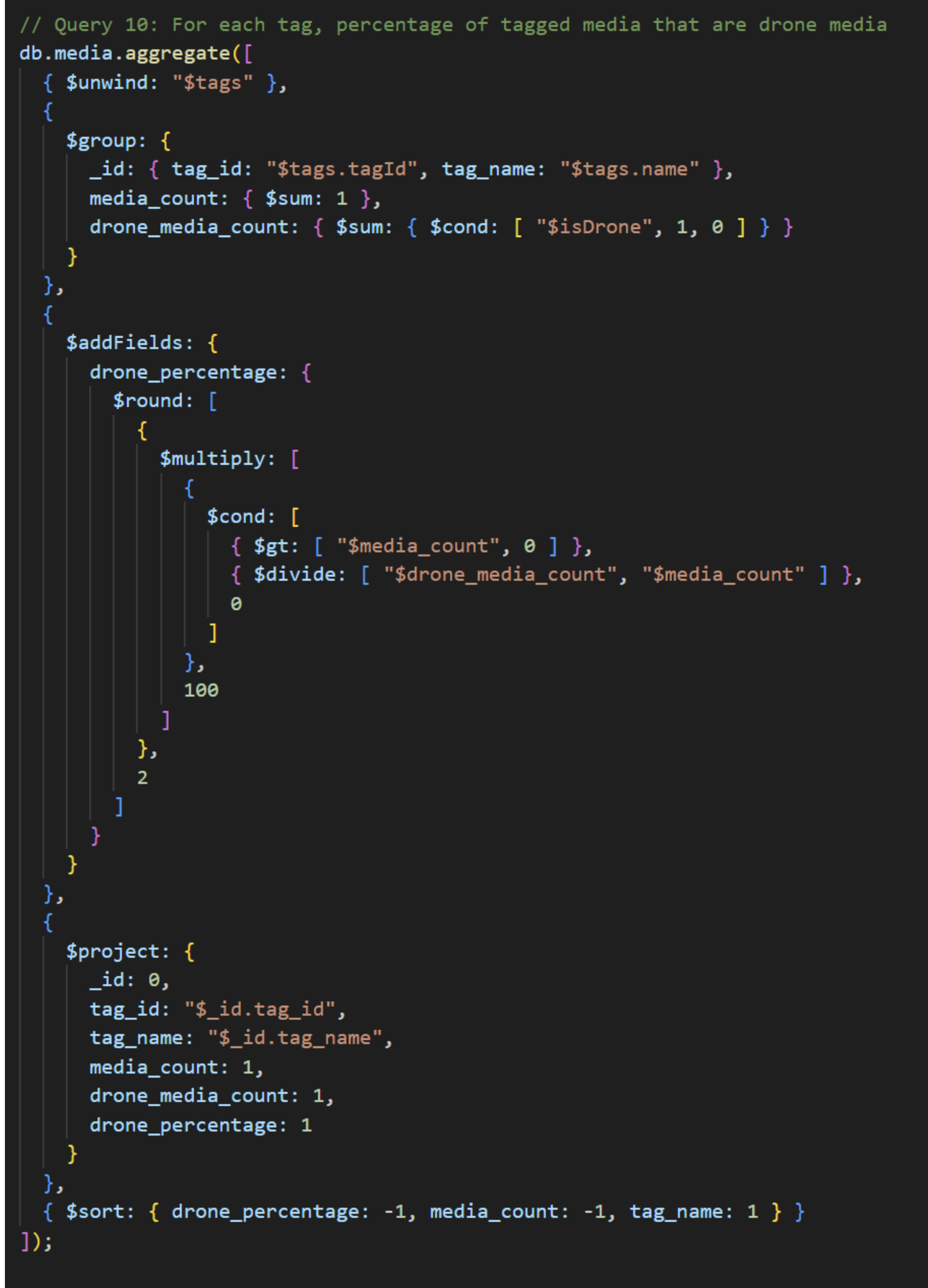
```
// Query 2: Top 10 most-used tags across all media
db.media.aggregate([
  { $unwind: "$tags" },
  {
    $group: {
      _id: { tag_id: "$tags.tagId", tag_name: "$tags.name" },
      media_count: { $sum: 1 }
    }
  },
  {
    $project: {
      _id: 0,
      tag_id: "$_id.tag_id",
      tag_name: "$_id.tag_name",
      media_count: 1
    }
  },
  { $sort: { media_count: -1, tag_name: 1 } },
  { $limit: 10 }
]);
```



```
// Query 3: Collection item counts (total / drone / photo / video) per collection
db.media.aggregate([
  { $unwind: "$collections" },
  {
    $group: {
      _id: {
        collection_id: "$collections.collectionId",
        collection_name: "$collections.name"
      },
      total_items: { $sum: 1 },
      drone_items: { $sum: { $cond: [ "$isDrone", 1, 0 ] } },
      photo_items: { $sum: { $cond: [ { $eq: [ "$mediaType", "PHOTO" ] }, 1, 0 ] } },
      video_items: { $sum: { $cond: [ { $eq: [ "$mediaType", "VIDEO" ] }, 1, 0 ] } }
    }
  },
  {
    $project: {
      _id: 0,
      collection_id: "$_id.collection_id",
      collection_name: "$_id.collection_name",
      total_items: 1,
      drone_items: 1,
      photo_items: 1,
      video_items: 1
    }
  },
  { $sort: { total_items: -1, collection_name: 1 } }
]);
```



```
// Query 4: Photo counts per month for year 2025
db.media.aggregate([
  {
    $match: {
      mediaType: "PHOTO",
      "capture.dateTime": { $type: "date" }
    }
  },
  {
    $set: {
      capture_year: { $year: { date: "$capture.dateTime", timezone: "UTC" } },
      capture_month: { $month: { date: "$capture.dateTime", timezone: "UTC" } },
      month_name: {
        $dateToString: {
          format: "%M",
          date: "$capture.dateTime",
          timezone: "UTC"
        }
      }
    }
  },
  { $match: { capture_year: 2025 } },
  {
    $group: {
      _id: {
        capture_year: "$capture_year",
        capture_month: "$capture_month",
        month_name: "$month_name"
      },
      photo_count: { $sum: 1 }
    }
  },
  {
    $project: {
      _id: 0,
      capture_year: "$_id.capture_year",
      capture_month: "$_id.capture_month",
      month_name: "$_id.month_name",
      photo_count: 1
    }
  },
  { $sort: { capture_year: 1, capture_month: 1 } }
]);
```



```
// Query 10: For each tag, percentage of tagged media that are drone media
db.media.aggregate([
  { $unwind: "$tags" },
  {
    $group: {
      _id: { tag_id: "$tags.tagId", tag_name: "$tags.name" },
      media_count: { $sum: 1 },
      drone_media_count: { $sum: { $cond: [ "$isDrone", 1, 0 ] } }
    }
  },
  {
    $addFields: {
      drone_percentage: {
        $round: [
          {
            $multiply: [
              {
                $cond: [
                  { $gt: [ "$media_count", 0 ] },
                  { $divide: [ "$drone_media_count", "$media_count" ] },
                  0
                ]
              },
              100
            ]
          },
          2
        ]
      }
    }
  },
  {
    $project: {
      _id: 0,
      tag_id: "$_id.tag_id",
      tag_name: "$_id.tag_name",
      media_count: 1,
      drone_media_count: 1,
      drone_percentage: 1
    }
  },
  { $sort: { drone_percentage: -1, media_count: -1, tag_name: 1 } }
]);
```

Compare & Contrast: MySQLvs MongoDB

- Data model:
  - MySQL: tables, strict schema, joins through FKs.
  - MongoDB: single media collection with embedded documents, metadata and arrays.
- Ingestion & metadata
  - MySQL: import pipeline is fragile and high-effort when EXIF changes.
  - MongoDB: easier to accept partial or new fields without changing schema.
- Analytics
  - MySQL: concise SQL for relational questions (GROUP BY, JOIN).
  - MongoDB: powerful but more complex aggregation pipelines.
- Scalability for this use case
  - MySQL: fine for core gallery and reporting, but needs automation around ingestion.
  - MongoDB: better fit for changing media metadata, especially if I automate imports.

What I Learned and Challenges

- Metadata challenges:
  - EXIF fields missing or inconsistent across tools (Windows vs Lightroom vs exports).
  - Deciding which fields deserve real columns vs staying optional.
- Displaying media in the application layer:
  - Keeping full-resolution, display-size, and watermarked paths straight.
  - Fixing file-path mismatches that caused broken images and wrong versions to show.
- Overall lessons:
  - Design from real data and workflows, not just an abstract schema.
    - Easy to make sample data. More difficult to manipulate and work with real data.
  - Ingestion and metadata quality are just as important as table design. They need to be exact and consistent.
  - Modeling the same database in both MySQL and MongoDB gives a deeper understanding of trade-offs.
    - MongoDB in my opinion is more difficult to start with, but once configured properly, it is much easier to use.

GitHub Repository

- Scan QR to open GitHub repository in browser.
- HTTPS clone:

  https://github.com/bcshort1/Database-Project.git
- GitHub Desktop Command-Line Interface clone:

  ```
  gh repo clone bcshort1/Database-Project
  ```