Lab 3 - Document databases

- **Due** Dec 12, 2023 by 11:59pm
- **Points** 100
- **Submitting** a file upload

For this assignment, you will use MongoDB to retrieve and analyze data stored in various MongoDB *Document Databases*.

The lab has three sections, each emphasizing different MongoDB capabilities and/or databases.

Lab prerequisites

You need access to the MongoDB sample databases (Canvas overview, MongoDB docs) to do this lab. You are welcome to use the shared Atlas database server we are using for this class (setup instructions), or to set up an AtlasDB server of your own, loaded with the default MongoDB sample datasets. If you are using your own datasets, do not modify them from their original state to answer your queries. If you have modified the sample databases, please use the shared Atlas server to answer the lab questions.

Although I strongly encourage you to use Studio3T as your GUI for working with MongoDB, you can use another client to construct and test your queries.

Answer format:

Please put your answers to each question into a **text** fileLinks to an external site. (.txt or .js). You must submit them using raw text, not formatted text (as you would create with MS Word, Google Docs, or PDF files). If you submit your answers in a formatted file, the formatting information invisibly added by MS Word (or Google Docs or Apple's Pages) will often break your queries when I copy them and try to run them for testing, which makes them wrong. Which does not lead to a good grade for you. Besides, if you want to be a Data Ninja, you'll not just have to learn to use a text editor; you'll need to develop strong opinions about why Text Editor A is better than

Text Editor B. So find one that you like and learn to wield it as the tool and weapon it is (pro tip - if you're using a Mac, BBEditLinks to an external site. is the best text editor available. Disagree? Those are fightin' words :-) See? Strong opinions about this topic are expected of data ninjas.

Use the following template to answer each question:

```
-- -------------------------
-- Question #
-- -------------------------

MongoDB query you used to compute the answer to this question
```

As shown in this template, it is much easier to grade your lab if you put a clear header above each question. Please do so.

For most questions, you will be given a precise spec regarding the format of the return documents. You need to match that format in your result set, though **the order of the keys does not matter because you cannot control the order in which MongoDB returns keys *inside* a document**. But returning *precisely* the specified keys in *precisely* the specified document structure is an important part of the task. Returning extra fields (including _id), missing requested fields, or returning incorrectly named fields will all result in points being deducted.

Section 1 *- Shipwrecks dataset. For the first three questions, you must write queries to answer the following questions about the shipwrecks stored in the sample_geospatial dataset. These are designed to be relatively straightforward queries in a simple dataset. You should only need to find the documents specified and transform/format them as requested. You may use either aggregation pipelines or find() to answer the first three questions. Your call.*

Question 1: Write a query that determines the total number of wrecks located north of the Arctic Circle. You may need to do some research to determine how you would determine whether a wreck is located north of the Arctic Circle. That research is part of the question. Your query should not return details about the matching wrecks, just an integer value indicating how many of them the dataset contains.

Question 2: Some shipwrecks are more dangerous than others. Some warships, for example, may have sunk carrying explosives. Helpfully, those are marked in this dataset by having their "history" key set to the value "Loaded with explosives".

Write a query that retrieves the location of all wrecks "Loaded with explosives". Each document in your result set should include the id, chart, history, water level (watlev), depth, and coordinates of one such plane wreck. The documents returned by your query need to match the following sample document structure (remember that you cannot control the order of keys listed within a document!):

```
{
    "_id": ...,
    "chart" : ...,
    "depth" : ...,
    "history" : ...,
    "watlev": ...,
    "coordinates" : [
        ...,
        ...
    ]
}
```

Question 3: Write a query that retrieves all shipwrecks that are categorized as "Wrecks - Visible" ("feature_type" field), with a water level description (watlev) of "covers and uncovers", at a latitude between 45.0 and 60.0 degrees (inclusive). In the northern hemisphere, latitude values increase from the equator, which is at 0.0 degrees latitude, to the north pole, which lies at 90.0 degrees latitude.

Your query should return the list of wrecks as a set of documents with the following format. The source field name is provided in comments after the result set field name if the source field to use is not obvious.

```
{
    "_id" : ObjectId("..."),
    "Wreck Coordinates" : [    // sample coordinates given, obviously they will va
ry from wreck to wreck
        -151.346957,
        60.516213
    ],
    "Wreck Status" : ...,  // map the "feature_type" field in the source data to t
his field name in the result set
    "Water Level" : ...  // map the "watlev" field in the source data to this fiel
d name in the result set
}
```

Section 2 - *Use the sample_airbnb dataset to answer the following questions.*

The sample_airbnb dataset is filled with data scraped from Airbnb's website (www.airbnb.comLinks to an external site.) for a handful of markets worldwide. This is a small sample of all Airbnb data. Review the Airbnb website as needed to understand the context for the data contained in this dataset.

You need to create aggregation pipelines to answer the remaining questions in the lab -- db.collection.aggregate([...]).

This section focuses on creating more sophisticated aggregation pipelines and using additional operators beyond just the $match and $project stages required in the first group of questions.

Question 4: Write an aggregation pipeline that determines how many listings the database contains for rentals in the "Bondi Beach" suburb of the Sydney market ("address.market"), that have at least two bedrooms and include the amenities "Kitchen" and "Air conditioning".

Your query should be in the form of an aggregation pipeline that returns one document containing a single key named "Matches found". The value of the "Matches found" key should be the total number of listings in this dataset that meet the specified criteria.

```
    {
       "Matches found" : ...
    }
```

Question 5: Now write an aggregation pipeline that retrieves the three rentals in the "Bondi Beach" suburb of Sydney that have the highest nightly rental price. Your result set should be sorted from the most expensive listing down to the third most expensive one. Your results should only include the three most expensive rentals in the Bondi Beach suburb.

Your query should return a set of three documents, each of which has the following structure:

```
{
   "Listing URL" : ...,
   "Listing name" : ...,
   "Nightly price" : ...,
   "Suburb": ... // from the "address.suburb" field
   "Airbnb market": ..., // from the "address.market field
}
```

Question 6: Write an aggregation pipeline to calculate the number of listings, min, max, and average nightly rental price for each of the following Hawaiian markets ("address.market"): Oahu, Kauai, Maui, and "The Big Island".

Your aggregation pipeline should calculate the total number of listings available in each of these markets, along with a nightly pricing summary.

Your result set should contain a set of documents with the following structure for each Hawaiian sub-market. Order your results from the market with the most listings down to the market with the fewest listings.

```
   {
     "Hawaiian Market": ...,
     "Number Of Listings": ...,
     "Lowest nightly listing price": ...,
     "Average nightly listing price": ...,
```

```
    "Highest nightly listing price": ...,
  }
```

Question 7: Let's switch from looking at vacation destinations to exploring how the Airbnb rental inventory in various cities matches the needs of digital nomads traveling on their own.

To do so, write an aggregation pipeline that finds the number of listings and the average nightly price for listings in each of these cities that meet the following criteria:

- Markets to include: New York, Istanbul, Barcelona, Hong Kong, Porto, Sydney
- One-bedroom listings only
- No roommates or sharing - room type needs to be "Entire home/apt"
- Amenities must include Wifi, Kitchen, and Coffee maker
- The overall review rating is 95 or higher (review_scores.review_scores_rating)

For each of these cities, calculate the following and return the results in a set of documents with the following structure:

```
{
   "City" : ...,     // use address.market
   "Number Of Listings" ...,
   "Average nightly price" : ...
}
```

Order the results of your query from the city with the fewest rentals meeting these criteria to the city with the most listings that meet these criteria.

Section 3 *- Use the sample_mflix dataset to answer the remaining questions.*

The sample_mflix dataset appears to be built from information about films and theaters gathered from imdb.comLinks to an external site. and rottentomatoes.comLinks to an external site.. Feel free to use those sites for reference when exploring the document structure for the movies listed in sample_mflix.

As with the previous section, all of your answers to questions in this section should be answered by creating an aggregation pipeline. This section emphasizes working with arrays and subdocuments.

Question 8: Write an aggregation pipeline that calculates the total number of theaters in each zipcode in the sample_mflix.theaters collection. Return a list of documents for all zip codes with more than five theaters. For each zip with more than five theaters, return a document with the following structure:

```
{
   "Zip" : ...,
   "TotalTheaters" : ...
}
```

Sort your result set in numeric order downwards from the Zip with the most theaters to the Zip with the fewest (but still more than 5).

Question 9: Write a query that calculates the number of films released between 2003 and 2013 (inclusive) that were directed by each of the following people.

- Martin Scorsese
- Christopher Nolan
- Tim Burton
- Spike Lee
- Lucy Walker
- Peter Jackson
- Susanne Bier
- Sang-soo Hong
- Robert Rodriguez

Use the field "year" to determine when each film was released.

Each of the documents returned in your result set should have the following structure:

```
{
   "Director" : ...,
   "TotalFilmsDirected" : ...
}
```

Order your result set from the director who released the most movies during that period down to the director who released the fewest.

Question 10: Write an aggregation pipeline that returns the names of all actors who appeared in more than ten Comedy films released between 2008 and 2016 (inclusive). A movie is classified as a Comedy in the sample_mflix movies dataset if the value "Comedy" appears in its list of genres. Use the field "year" to determine the year a film was released.

For each actor who meets these criteria, your query should return the actor's name, the total number of Comedy films they appeared in during that time, and the average IMDb rating of those films.

Order your results according to the average IMDb rating for each actor's comedy films, highest to lowest.

Your result set should include documents with the following structure:

```
{
    "Actor" : ...
    "NumberOfComediesReleased": ...,
    "AverageIMDBRating" : ...,
}
```

Remember that you cannot control the order in which key:value pairs are displayed within a JSON document.

Deliverables and submission instructions:

Submit a text file (.txt or .js) containing your answers to the lab questions.

To grade your lab, I will copy and paste your queries from this file to test them on the shared MongoDB Atlas instance. Submissions of query statements and results submitted in plain text format are much easier to grade in this manner, as cutting and pasting from an MS Word or PDF document often introduces odd (and sometimes invisible) characters that can mess up the query, causing it to either fail outright or to return the wrong values, neither of which are good outcomes for grading your assignment. So please submit your queries and results in a plain text file with a .txt or .js suffix.

Grading Criteria:

Each of the questions you need to answer is worth 10 points. You will earn between 0 and 10 points for each question using the following rubric:

Points earned	Description
10	The answer is completely correct. The query returns precisely what was specified. P and interpretation of results is precisely as specified. No errors are present. If additio to be added to the dataset or some documents in the dataset were to be modified or r query would still return the correct results.
6-9	The query returns precisely what was specified, and the answer is materially correct, presentation and/or interpretation of results is either slightly incorrect or is done so t meet the specification in the assignment. Alternatively, the answer is materially corr details of the query, results, or conclusions drawn from them are not quite correct.
1-5	Results submitted are materially incorrect, but the query is, to a greater or lesser deg "directionally correct". Your score will be closer to 5 if the query is heading more or less in the right directi made a mistake, such as returning the wrong fields, filtering out a slightly incorrect s documents for your results, etc. This will typically be one mistake. Your score will be closer to 1 if the query makes more than one mistake and/or the r returned are obviously incorrect but submitted anyway. An example of an obviously answer might be submitting an answer that states that the average score of the winni

	given season was 676,642 points, even though a typical score is closer to 100 points That result is so far from correct that you should clearly catch it before submitting it
0	Results are materially incorrect, and the query submitted is not close to providing th results.

Statement on Collaboration:

This lab may be completed individually or in groups of up to three students. If you are working as a group, you should work with the same lab partners with whom you completed the previous labs. If, for some reason, you are unable to work with the same partners for lab 3, please contact me (Prof. Monroe) to explain the situation and what your changed working situation will be. I typically need to change Canvas groups in a certain way to avoid messing up previous and future grade assignments for group members.

You need to work with only your group on all issues specific to creating and debugging the queries assigned.

You are, however, free to discuss general issues about the assignment, working with MongoDB, and the structure of the datasets with your classmates on other teams, just not the specifics of the queries you need to write to answer the questions posed in the assignment. There is, unfortunately, not a particularly clear line to be drawn between acceptable collaboration and inappropriate sharing of results. Use your best judgment, and if you feel that after your discussion with a classmate, the two of you would likely come up with *exactly* the same approach to producing the required query, then you have probably done too much work together. If you find that you are just copying the ideas from another group, you have almost certainly stepped over the line that separates a constructive discussion about the assignment from cheating. If you find yourself in either of these situations, talk with the instructor before submitting your report so that we can find an appropriate resolution to the problem before it becomes an academic integrity issue.

If your team has spent so much time discussing the assignment with another team that the instructor will likely be concerned about the similarity of your submissions, then you should list the names of the other people in the class with whom you have discussed the assignment as well. These people should be identified as participants in your discussions rather than team members. Those identified on an assignment only as participants will not be graded or get credit for that team's submission but they will also be much less likely to get in trouble for academic integrity problems if their participation is called out clearly in the submission.