University of Salford, MSc Data Science

Module: Big Data Tools and Techniques

Date: Trimester 2, 2024-2025

Session: Workshop Week 9

Topic: MongoDB

Tools: Jupyter Notebook and MongoDB Atlas

Instructors: Dr Kaveh Kiani, Dr Taha Mansouri, and Nathan Topping.

Objectives:

After completing this workshop, you will be able to:

- Prepare a MongoDB Atlas account
- Implement queries and aggregations on Atlas
- Connect to and interact with Atlas through Python
- Design a pipeline to process data

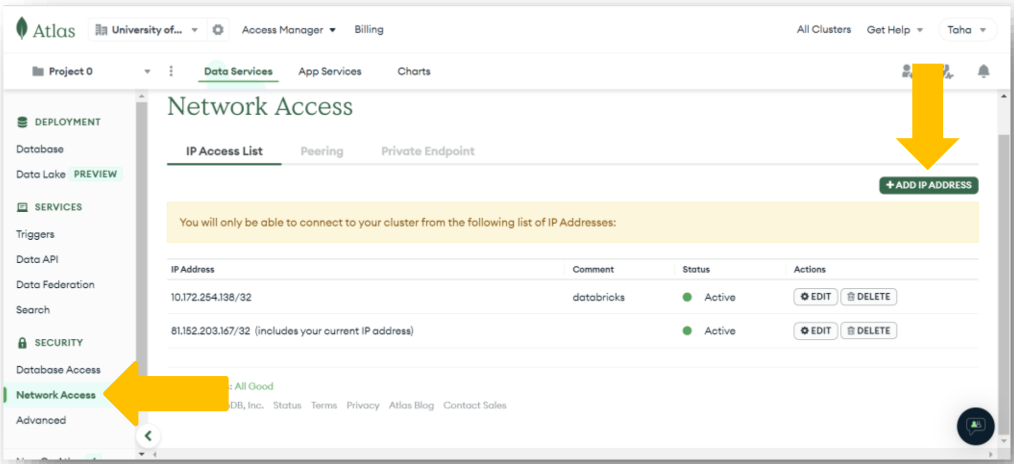
Table of Contents

Part 1: Fire up the Atlas workspace

1- Login to your Atlas free version through the link below:

https://account.mongodb.com/account/login?signedOut=true

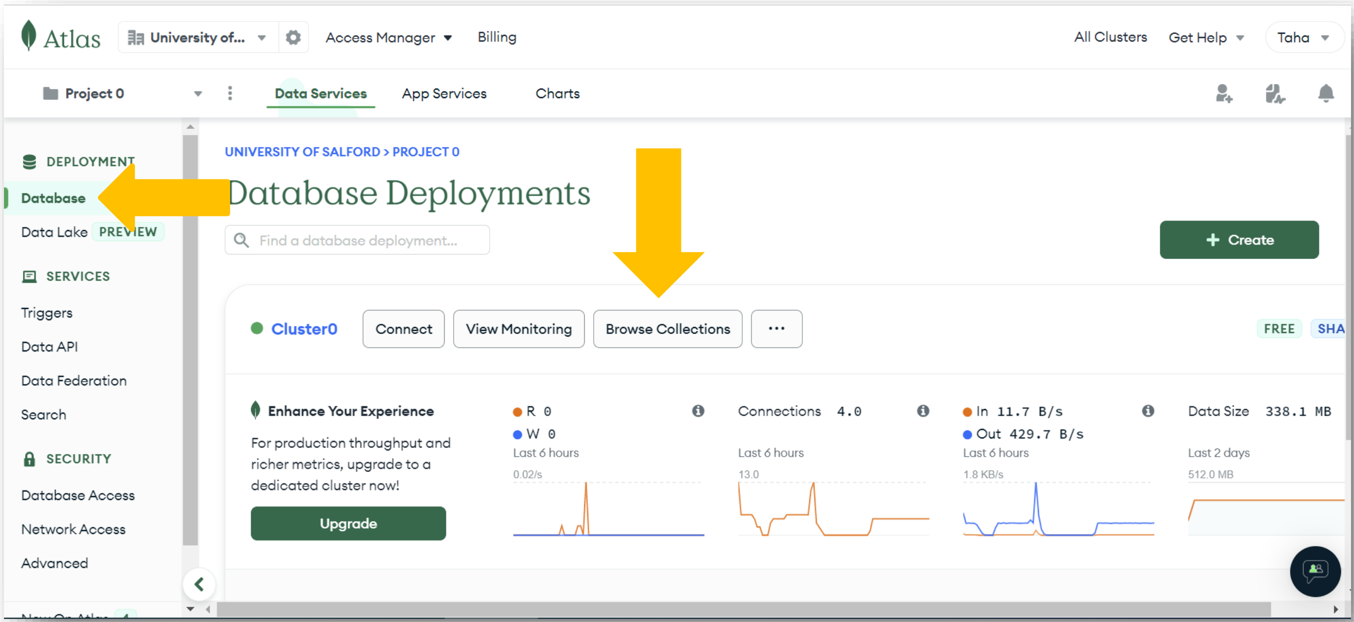
2- Make sure your current IP is listed as the trusted IP. If not, go through Network Access and add your local IP address.



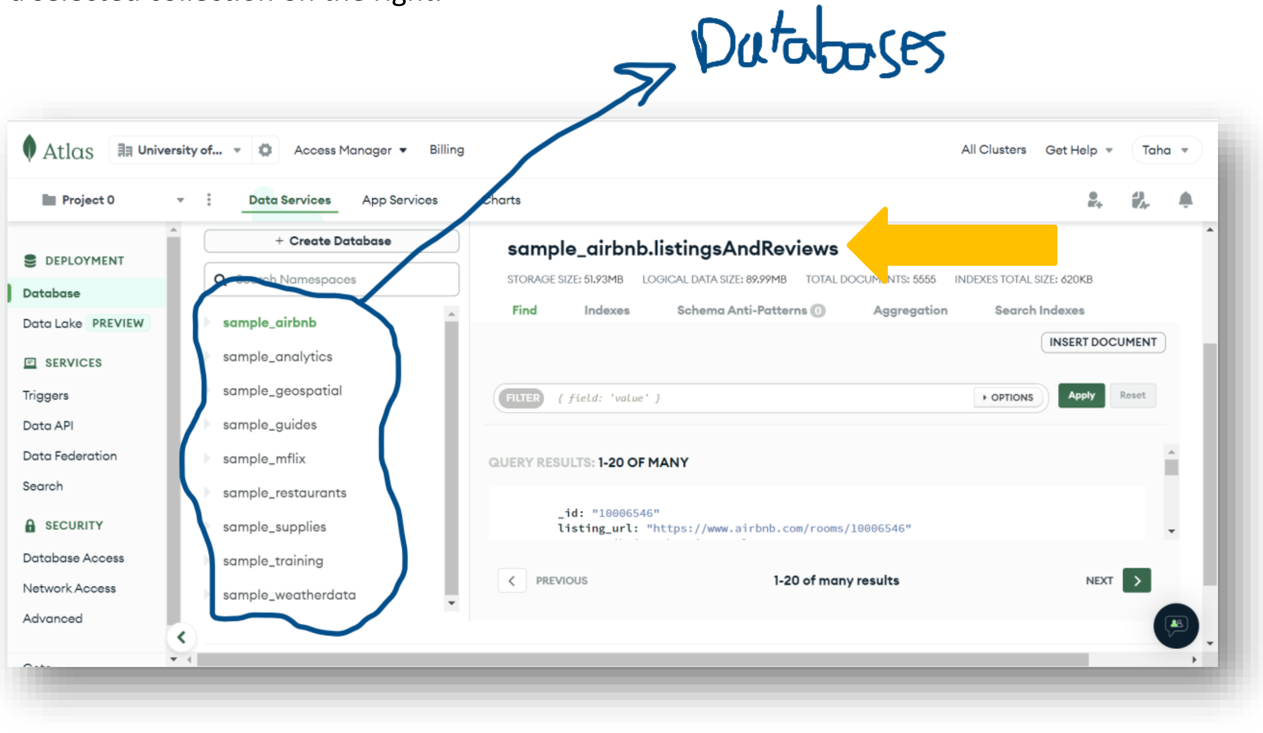
You will only be able to connect to your cluster from the following list of IP Addresses:

IP Address	Comment	Status	Actions
10.172.254.138/32	databricks	● Active	EDIT DELETE
81.152.203.167/32 (includes your current IP address)		● Active	EDIT DELETE

3- Click on the Database tab.



4- Click on Browse Collections. Now you can see the list of databases on the left, and the content of a selected collection on the right.



```
_id: "10006546"
listing_url: "https://www.airbnb.com/rooms/10006546"
```

Part-2: MongoDB

MongoDB is a cross-platform, document-oriented NoSQL database. It is designed to store and manage unstructured or semi-structured data. Unlike traditional relational databases, MongoDB uses a flexible document model, which allows developers to store and query data in a more intuitive and natural way. Data in MongoDB is stored in documents, which are similar to JSON objects and can contain any number of fields, arrays, and sub-documents. This makes it easy to store complex data structures and to modify them as requirements change.

MongoDB stores data records as documents (specifically BSON documents) which are gathered in collections. You can create secondary indexes on these collections, join them together, and use the powerful aggregation framework embedded in MongoDB. A database stores one or more collections of documents.

In this workshop we mostly work with reading data from MongoDB. To select all documents in the collection, pass an empty document as the query filter parameter to the find method. The query filter parameter determines the select criteria:

```
db.inventory.find( {} )
```

This operation uses a filter predicate of {}, which corresponds to the following SQL statement:

```
SELECT * FROM inventory
```

To specify equality conditions, use <field>:<value> expressions in the query filter document:

```
db.inventory.find( { status: "D" } )
```

The following example selects from the inventory collection all documents where the status equals "D":

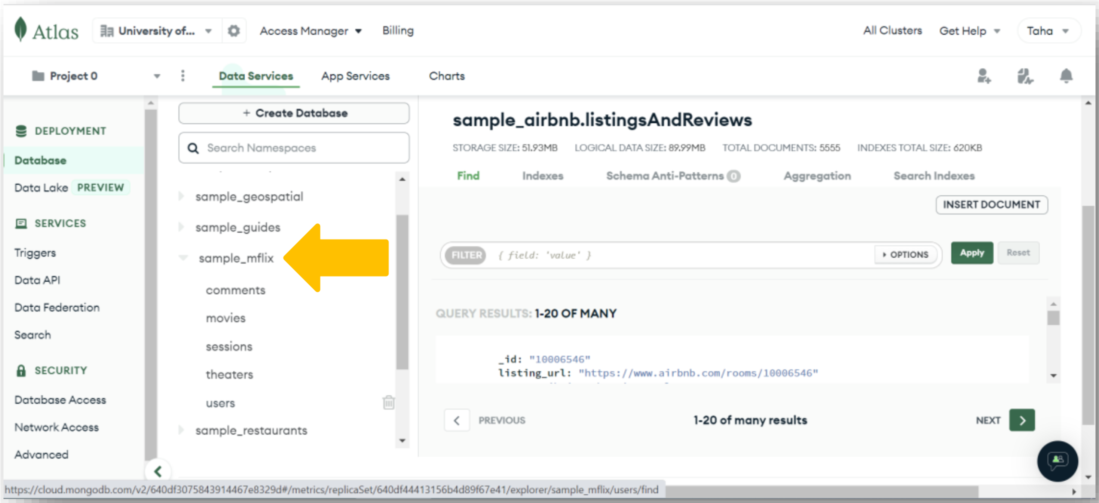
```
SELECT * FROM inventory WHERE status = "D"
```

This operation uses a filter predicate of { status: "D" }, which corresponds to the following SQL statement:

```
{ <field1>: <value1>, ... }
```

Part-3: Working with Atlas

1- On Atlas, expand sample_mflix database



sample_airbnb.listingsAndReviews

STORAGE SIZE: 51.93MB LOGICAL DATA SIZE: 89.99MB TOTAL DOCUMENTS: 5555 INDEXES TOTAL SIZE: 620KB

```
{ field: 'value' }
```

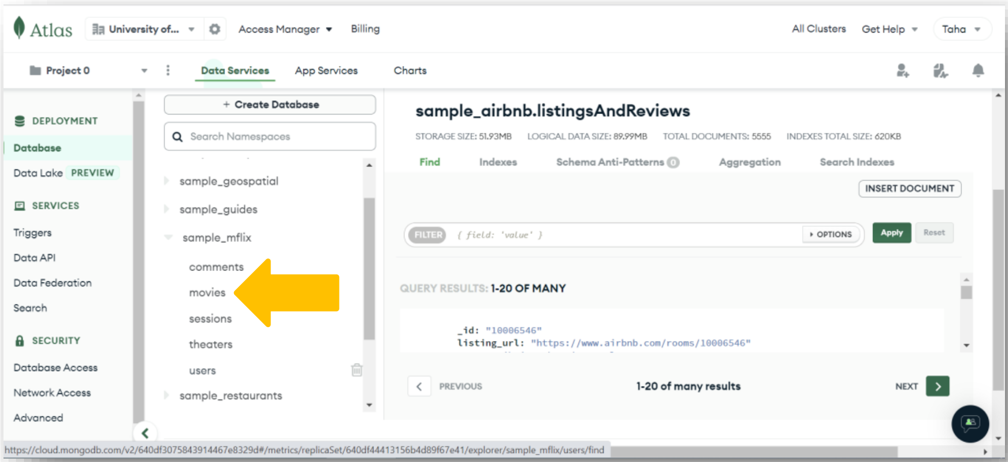
QUERY RESULTS: 1-20 OF MANY

```
_id: "10006546"
listing_url: "https://www.airbnb.com/rooms/10006546"
```

1-20 of many results

https://cloud.mongodb.com/v2/640df3075843914467e8329d#/metrics/replicaSet/640df44413156b4d89f67e41/explorer/sample_mflix/users/find

2- Select movies collection



sample_airbnb.listingsAndReviews

STORAGE SIZE: 51.93MB LOGICAL DATA SIZE: 89.99MB TOTAL DOCUMENTS: 5555 INDEXES TOTAL SIZE: 620KB

```
{ field: 'value' }
```

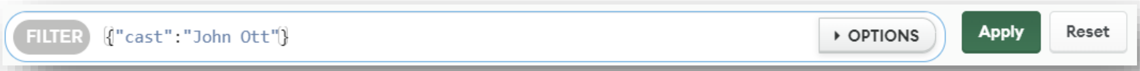
QUERY RESULTS: 1-20 OF MANY

```
_id: "10006546"
listing_url: "https://www.airbnb.com/rooms/10006546"
```

1-20 of many results

https://cloud.mongodb.com/v2/640df3075843914467e8329d#/metrics/replicaSet/640df44413156b4d89f67e41/explorer/sample_mflix/users/find

3- Filter the movie cast by "Brad Pitt"

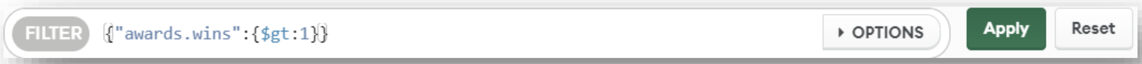


```
{"cast":"John Ott"}
```

4- Search for those movies that have won more than one award.

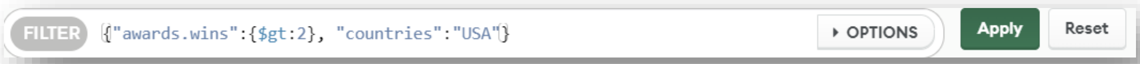
Note: To access a field inside of a nested document, you can use the dot operator.

Note: To use conditional operators such as greater than, less than, greater than and equals and so on, you can use an operator along with the intended value as a dictionary. For example, to check whether the value of a field is less than or equal to 5, you can use {“field_Name”: {$lte:5}}



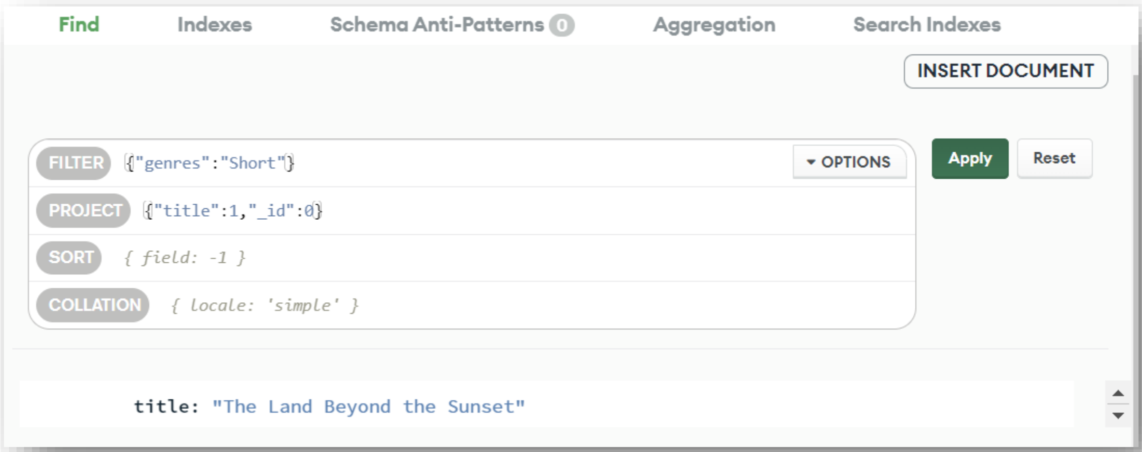
```
{"awards.wins":{$gt:1}}
```

5- Find those movies that have won more than one award and that have the USA as their country.



```
{"awards.wins":{$gt:2}, "countries":"USA"}
```

6- You can also specify a projection list to just show the information needed. Find those movies whose genre is “Short” and just project the title:



```
FILTER     {"genres":"Short"}
PROJECT    {"title":1,"_id":0}
SORT       { field: -1 }
COLLATION  { locale: 'simple' }

title: "The Land Beyond the Sunset"
```

Part-4: Aggregation Framework

The Aggregation Framework in MongoDB is a powerful data processing tool that allows you to perform complex data analysis on collections of documents in a database. It provides a set of operators that can be used to perform data filtering, grouping, sorting, and data transformations. With the Aggregation Framework, you can combine data from multiple collections, perform calculations on data, and analyse data in real-time. This makes it a very powerful tool for business intelligence and data analysis applications.

Some of the key features of the Aggregation Framework in MongoDB include:

- Pipelined data processing: The aggregation framework allows you to combine multiple operators into a single pipeline, where the output of one operator is the input to the next operator. This makes it easy to perform complex data transformations and analysis.
- Extensive operator set: The Aggregation Framework provides a wide range of operators for data filtering, grouping, sorting, and transformations. This includes operators for conditional logic, arithmetic operations, string manipulation, date manipulation, and more.
- Integration with MongoDB: The Aggregation Framework is tightly integrated with MongoDB, which means that it can take advantage of MongoDB's scalability, replication, and sharding features.

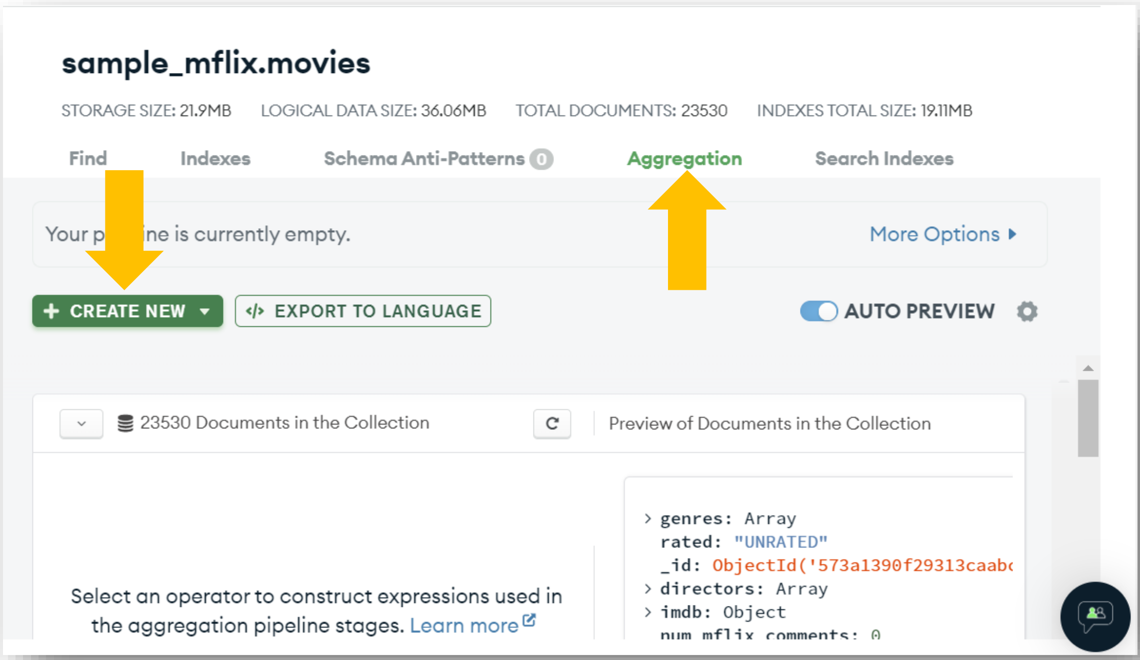
The MongoDB Aggregation Framework includes several stages that can be used to perform various data processing operations. The stages are applied in a pipeline, where the output of one stage becomes the input of the next stage. The stages in the Aggregation Framework are:

- $match: This stage is used to filter documents based on certain criteria. It works like a query filter and can use various comparison operators to filter documents.
- $project: This stage is used to select certain fields from documents and project them in the output. It can also be used to create new fields or transform existing fields.
- $group: This stage is used to group documents by a specified field or fields. It can also perform various aggregate functions such as sum, average, and count on the grouped data.
- $sort: This stage is used to sort the output documents based on one or more fields. It can sort in ascending or descending order.
- $limit: This stage is used to limit the number of documents returned in the output.
- $skip: This stage is used to skip a specified number of documents in the input before processing.

- $unwind: This stage is used to break up an array field into separate documents, each containing a single value from the array.
- $lookup: This stage is used to perform a left outer join between two collections.
- $facet: This stage is used to perform multiple aggregation operations on the same set of input documents. It returns multiple sets of documents, each representing the result of a separate aggregation operation.

These stages can be combined in different ways to perform a wide range of data processing operations on MongoDB collections.

1- Click on movies collection and select the aggregation tab. Then click on "create new".



sample_mflix.movies

STORAGE SIZE: 21.9MB LOGICAL DATA SIZE: 36.06MB TOTAL DOCUMENTS: 23530 INDEXES TOTAL SIZE: 19.11MB

Your p ine is currently empty.

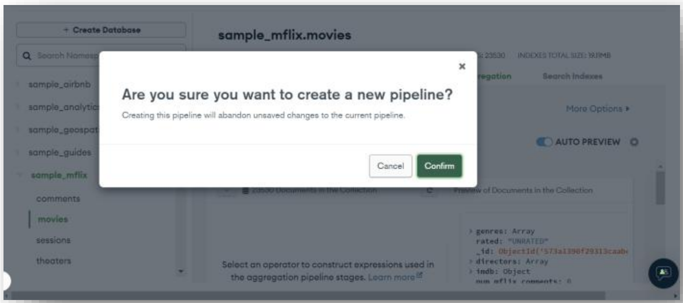
23530 Documents in the Collection

Preview of Documents in the Collection

```
> genres: Array
  rated: "UNRATED"
  _id: ObjectId('573a1390f29313caab
> directors: Array
> imdb: Object
  num_mflix_comments: 0
```

Select an operator to construct expressions used in the aggregation pipeline stages. Learn more

2- Another window will pop up, select confirm.

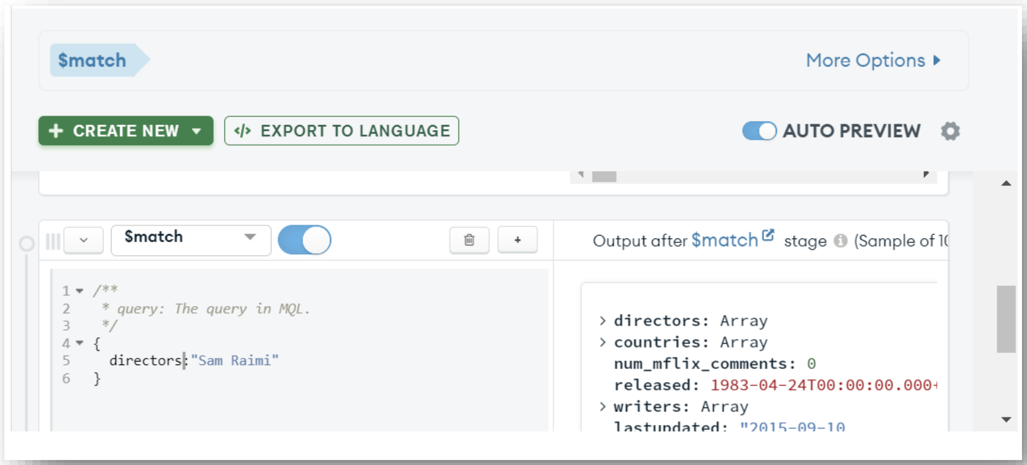


sample_mflix.movies

Are you sure you want to create a new pipeline?

Creating this pipeline will abandon unsaved changes to the current pipeline.

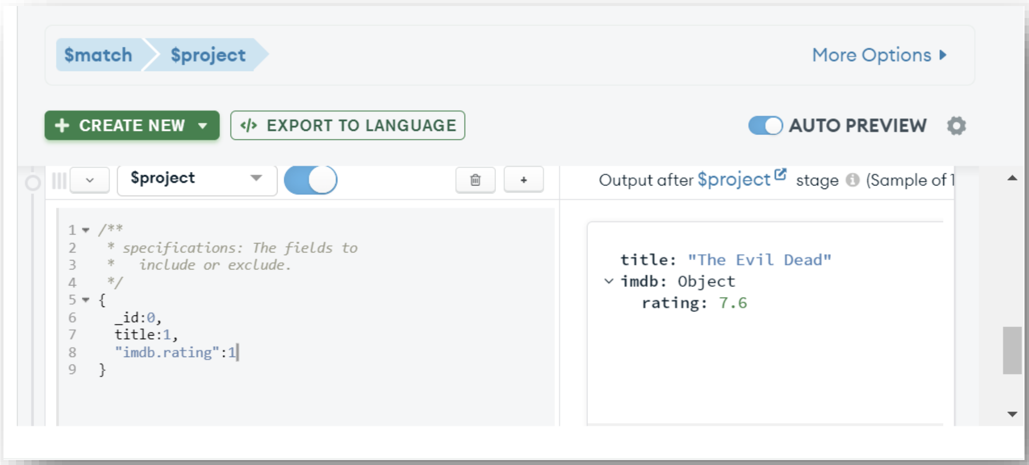
3- From the stages drop down list select $match and specify directors as “Sam Raimi”.



```
/**
 * query: The query in MQL.
 */
{
  directors:"Sam Raimi"
}
```

```
> directors: Array
> countries: Array
  num_mflix_comments: 0
  released: 1983-04-24T00:00:00.000
> writers: Array
  lastupdated: "2015-09-10
```

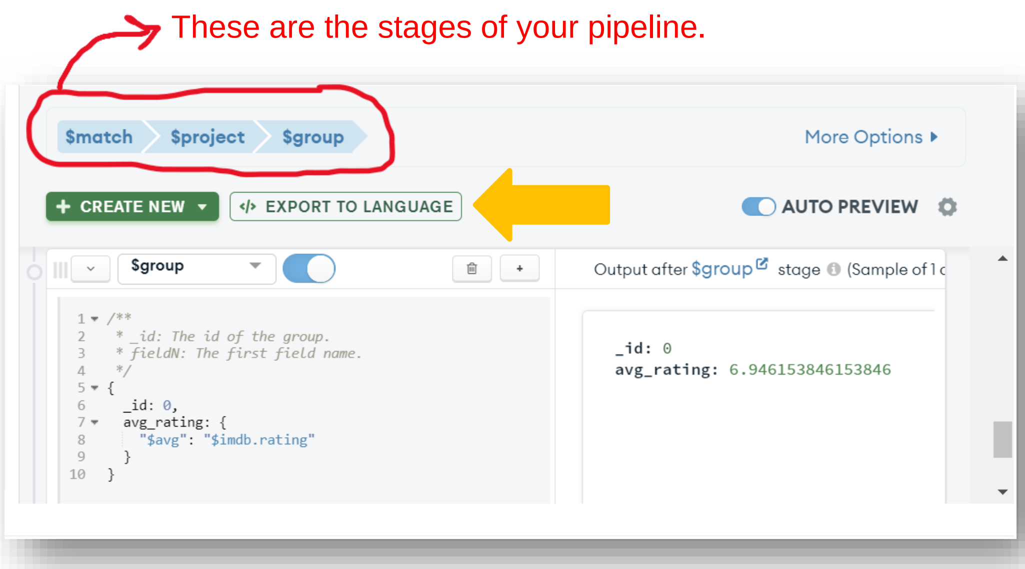
4- Add another stage and select its type as $project. Then filter out _id and select title and imdb rating to show.



```
/**
 * specifications: The fields to
 *   include or exclude.
 */
{
  _id:0,
  title:1,
  "imdb.rating":1
}
```

```
title: "The Evil Dead"
imdb: Object
  rating: 7.6
```

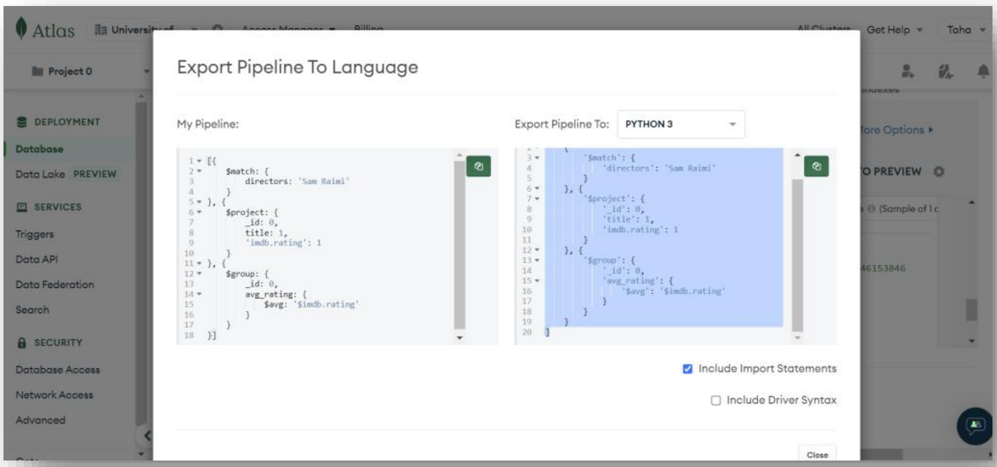
5- Add another stage and select $group. The objective is to calculate the average imdb rating for those movies that are directed by “Sam Raimi”.


These are the stages of your pipeline.
$match
$project
$group
More Options
CREATE NEW
EXPORT TO LANGUAGE
AUTO PREVIEW
$group
Output after $group stage (Sample of 1 d

```
/**
 * _id: The id of the group.
 * fieldN: The first field name.
 */
{
  _id: 0,
  avg_rating: {
    "$avg": "$imdb.rating"
  }
}
```

```
_id: 0
avg_rating: 6.946153846153846
```

6- By clicking on “EXPORT TO LANGUAGE”, you can export the pipeline into other programming languages.



My Pipeline:

```
[{
    $match: {
        directors: 'Sam Raimi'
    }
}, {
    $project: {
        _id: 0,
        title: 1,
        'imdb.rating': 1
    }
}, {
    $group: {
        _id: 0,
        avg_rating: {
            $avg: '$imdb.rating'
        }
    }
}]
```

Export Pipeline To: PYTHON 3

```
        '$match': {
            'directors': 'Sam Raimi'
        }
    }, {
        '$project': {
            '_id': 0,
            'title': 1,
            'imdb.rating': 1
        }
    }, {
        '$group': {
            '_id': 0,
            'avg_rating': {
                '$avg': '$imdb.rating'
            }
        }
    }
]
```

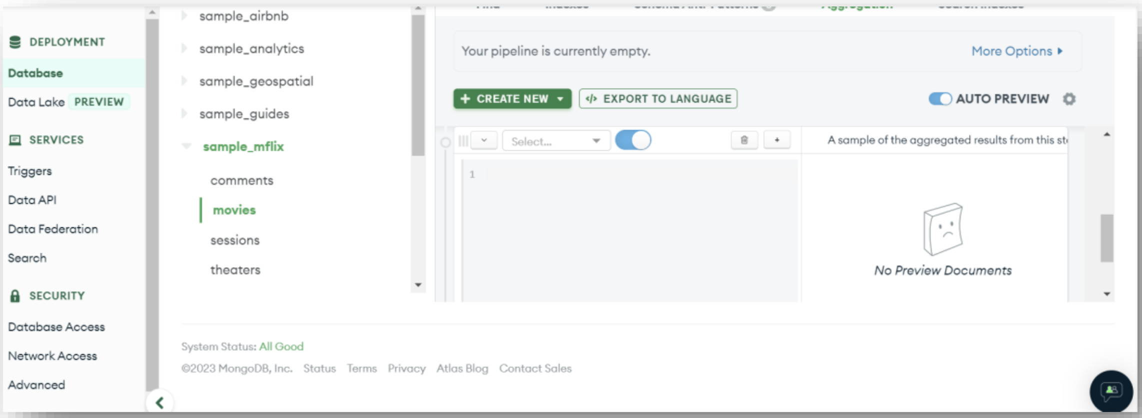
☑ Include Import Statements

☐ Include Driver Syntax

You can join some collections through their shared keys. To this end, you need to use $lookup stage.

The activity is to count the number of comments for each movie. There are two different collections as movies and comments. You need to join them together by movie_id in the comments collection and _id in the movies collection. We want to take 1980s movies into account.

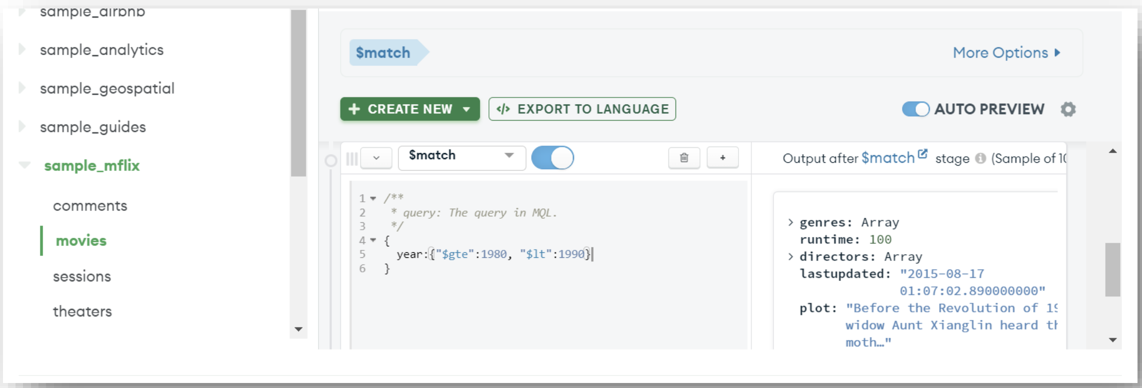
7- Select movies collection and the aggregation tab.



Your pipeline is currently empty.

A sample of the aggregated results from this st

8- In the current stage select $match and filter year between 1980 and 1990.

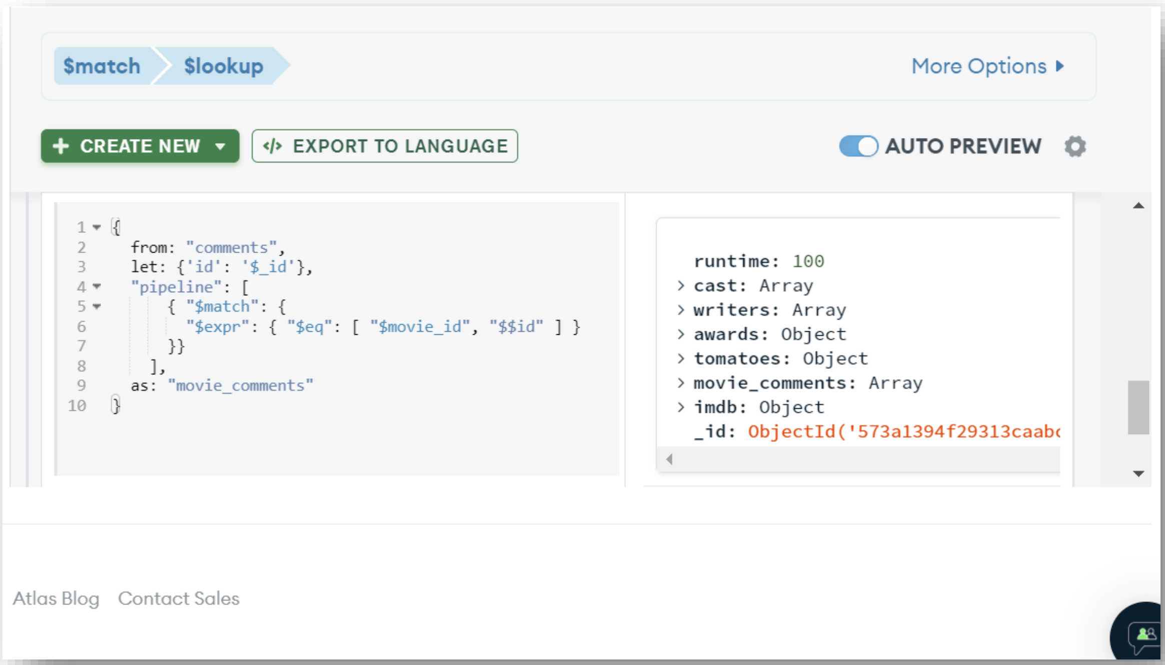


```
/**
 * query: The query in MQL.
 */
{
  year:{"$gte":1980, "$lt":1990}
}
```

Output after $match stage (Sample of 1

```
> genres: Array
  runtime: 100
> directors: Array
  lastupdated: "2015-08-17
               01:07:02.890000000"
  plot: "Before the Revolution of 1
        widow Aunt Xianglin heard th
        moth…"
```

9- Add a new stage and select $lookup as its type.

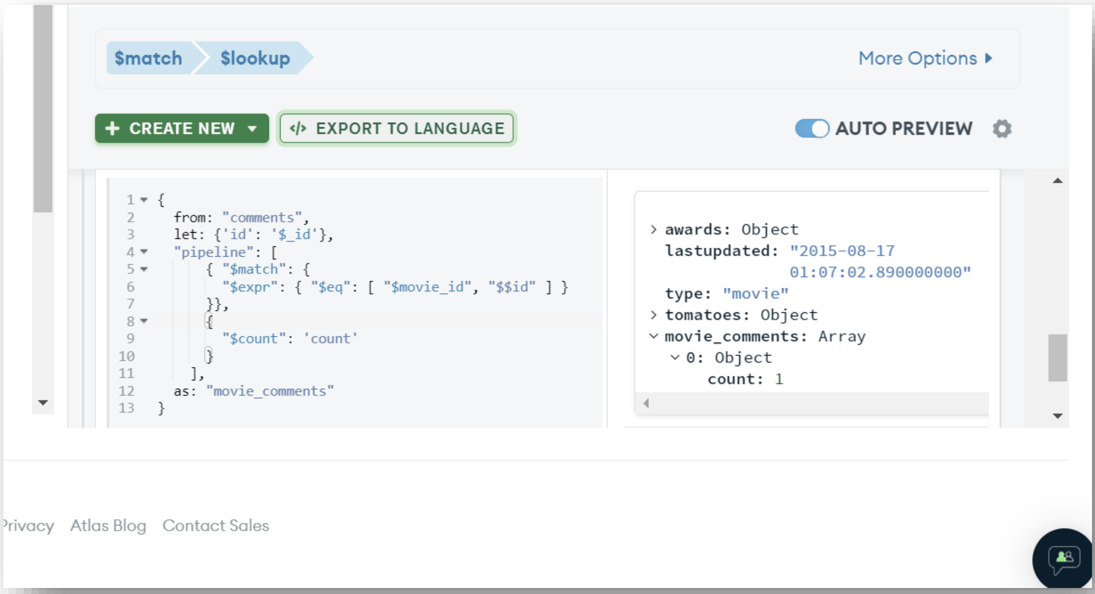


```
{
  from: "comments",
  let: {'id': '$_id'},
  "pipeline": [
    { "$match": {
      "$expr": { "$eq": [ "$movie_id", "$$id" ] }
    }}
  ],
  as: "movie_comments"
}
```

```
  runtime: 100
> cast: Array
> writers: Array
> awards: Object
> tomatoes: Object
> movie_comments: Array
> imdb: Object
  _id: ObjectId('573a1394f29313caab
```

Note: from shows the collection that you want to join to. let links the _id of movies collection to a temporary variable as id and make it accessible inside of the pipeline. In pipeline you can define any aggregation stages, however, you have to join the primary and foreign keys together. And finally, as defines the name of the desired field.

10- Add the $count stage inside of the $lookup pipeline. As it is a pipeline you don’t need to add any other stage.



```
{
  from: "comments",
  let: {'id': '$_id'},
  "pipeline": [
    { "$match": {
      "$expr": { "$eq": [ "$movie_id", "$$id" ] }
    }},
    {
      "$count": 'count'
    }
  ],
  as: "movie_comments"
}
```

```
> awards: Object
  lastupdated: "2015-08-17
               01:07:02.890000000"
  type: "movie"
> tomatoes: Object
v movie_comments: Array
  v 0: Object
      count: 1
```

Challenge-1

Calculate the average number of tomatoes viewer reviews of those movies that have a production year after 1920, English as their language, tomatoes viewer ratings which are greater than 3.5 and they have mflix comments.

Part-5: Working with MongoDB using Python

In this part we use Jupiter Notebook. To this end you need to install Anaconda on your system. Anaconda is the world's most popular data science platform, which helps users manage a collection of over 7,500+ open-source packages available to them. Anaconda Distribution equips individuals to easily search, install, and run thousands of Python/R packages and access a vast library of community content and support. It also makes creating, saving and loading programs very straightforward.

Installing Anaconda

Before installing Anaconda distribution, check the system requirements listed below:

System requirements:

• Operating system: Windows 8 or newer, 64-bit macOS 10.13+, or Linux, including Ubuntu, RedHat, CentOS 7+, and others.

(If your operating system is older than what is currently supported, you can find older versions of the Anaconda installers that might work for you on the Anaconda's archive page.)

• Minimum 5 GB disk space to download and install.

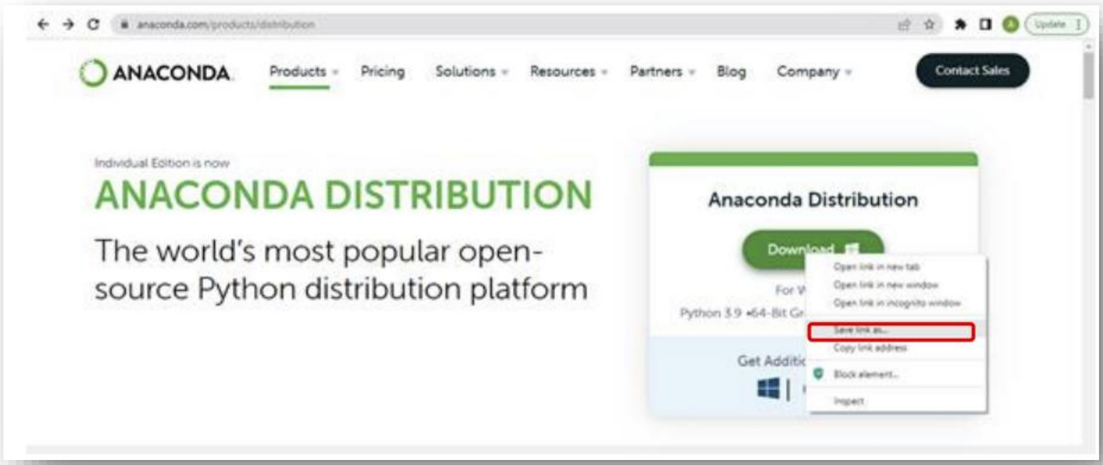
Downloading and installing on Windows:

You can download Anaconda installers from

https://www.anaconda.com/products/distribution

If you want to download Windows, Python 3.9, 64-Bit Graphical Installer, right click on the download button and click save link as. Save the Anaconda3-2022.05- Windows-x86_64.exe on your local computer (e.g., Downloads folder).

NB: You only need to carry out this step if you are using your own device and have not previously installed Anaconda. If you have previously installed Anaconda or are using a university device, please skip to page 18, step 1.



Individual Edition is now

ANACONDA DISTRIBUTION

The world's most popular open-source Python distribution platform

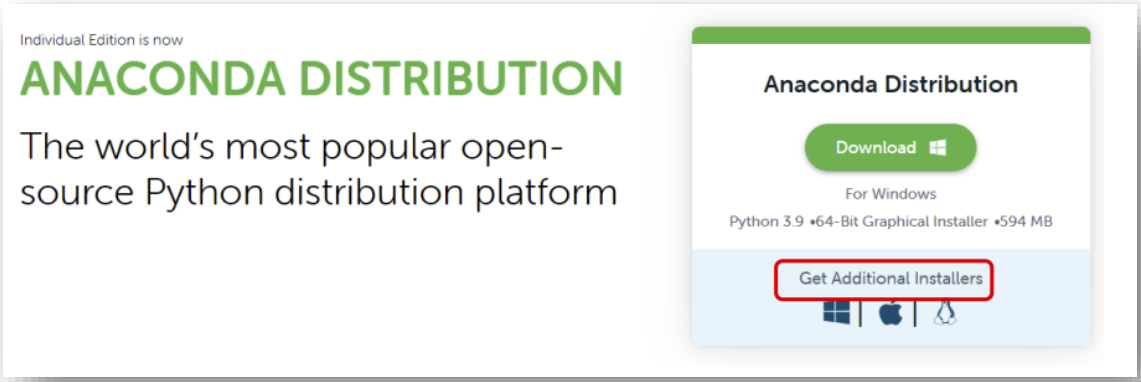
Anaconda Distribution

For W

Python 3.9 •64-Bit Gr

Get Additic

If you want to install Anaconda on other operating systems, click on "Get Additional Installer". This will take you to the bottom of the page where you can find other versions of the Anaconda installers. Right click on the option that works for you and click on "save link as" to download installer.



Individual Edition is now

ANACONDA DISTRIBUTION

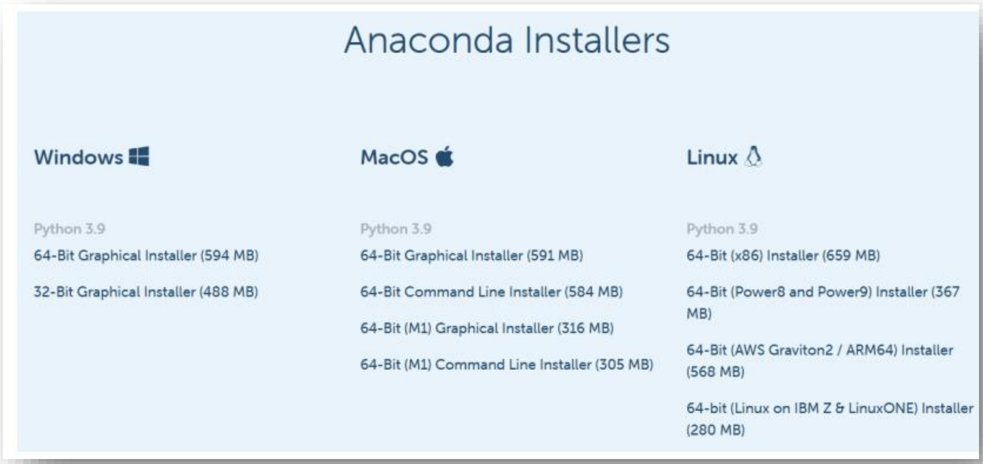
The world's most popular open-source Python distribution platform

Anaconda Distribution

For Windows

Python 3.9 •64-Bit Graphical Installer •594 MB

Get Additional Installers

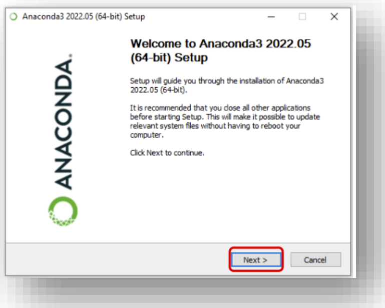


Anaconda Installers

Windows	MacOS	Linux
Python 3.9	Python 3.9	Python 3.9
64-Bit Graphical Installer (594 MB)	64-Bit Graphical Installer (591 MB)	64-Bit (x86) Installer (659 MB)
32-Bit Graphical Installer (488 MB)	64-Bit Command Line Installer (584 MB)	64-Bit (Power8 and Power9) Installer (367 MB)
	64-Bit (M1) Graphical Installer (316 MB)	64-Bit (AWS Graviton2 / ARM64) Installer (568 MB)
	64-Bit (M1) Command Line Installer (305 MB)	64-bit (Linux on IBM Z & LinuxONE) Installer (280 MB)

Go to your Downloads folder and double-click the installer to launch (If you encounter issues during installation, temporarily disable your anti-virus software during install, then re-enable it after the installation).

Click on the next button.



Anaconda3 2022.05 (64-bit) Setup

Welcome to Anaconda3 2022.05 (64-bit) Setup

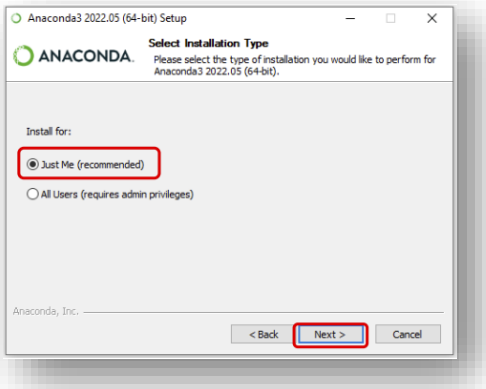
Setup will guide you through the installation of Anaconda3 2022.05 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Read the licensing terms and click on the "I Agree".

In the next step, choose "Just me" option and click on the Next button. (Only select an install for All Users if you need to install for all users' accounts on the computer. This requires Windows Administrator privileges).



Anaconda3 2022.05 (64-bit) Setup

Select Installation Type

Please select the type of installation you would like to perform for Anaconda3 2022.05 (64-bit).

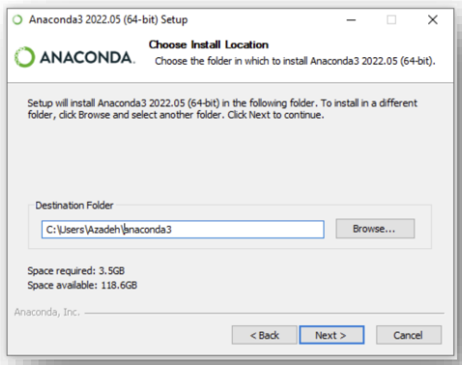
Install for:

● Just Me (recommended)

○ All Users (requires admin privileges)

Anaconda, Inc.

Select a destination folder to install Anaconda. You can use Browse button to change the location (The directory path should not contain spaces or unicode characters). After choosing install location click on the next button.


Anaconda3 2022.05 (64-bit) Setup

ANACONDA.

Choose Install Location

Choose the folder in which to install Anaconda3 2022.05 (64-bit).

Setup will install Anaconda3 2022.05 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

C:\Users\Azadeh\anaconda3

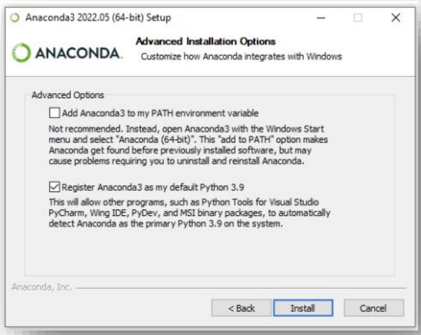
Browse...

Space required: 3.5GB
Space available: 118.6GB

Anaconda, Inc.

< Back | Next > | Cancel

In the next step, check “Register Anaconda3 as my default Python 3.9” and click on the Install button.


Anaconda3 2022.05 (64-bit) Setup

ANACONDA.

Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

☐ Add Anaconda3 to my PATH environment variable

Not recommended. Instead, open Anaconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

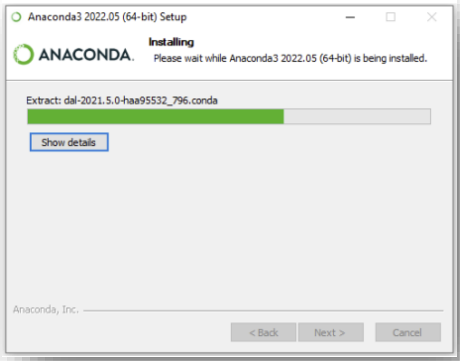
☑ Register Anaconda3 as my default Python 3.9

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.9 on the system.

Anaconda, Inc.

< Back | Install | Cancel

Please wait while Anaconda3 is being installed. It will take few minutes. Then the Next button will be enabled.


Anaconda3 2022.05 (64-bit) Setup

ANACONDA.

Installing

Please wait while Anaconda3 2022.05 (64-bit) is being installed.

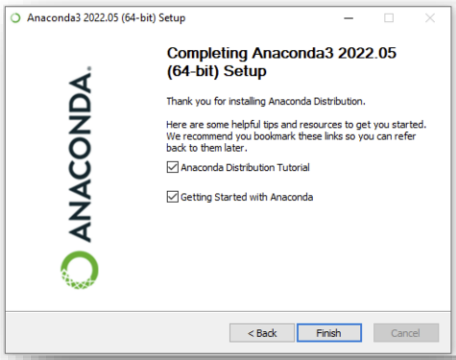
Extract: dal-2021.5.0-haa95532_796.conda

Show details

Anaconda, Inc.

< Back | Next > | Cancel

In the next Dialog box, click on the Next button. Finally, you should see the “Completing Anaconda3 Setup” dialog box. Click the Finish button to complete the installation (If you wish to read more about Anaconda.org and how to get started with Anaconda, check the boxes “Anaconda Distribution Tutorial” and “Getting started with Anaconda”.)


Anaconda3 2022.05 (64-bit) Setup

Completing Anaconda3 2022.05 (64-bit) Setup

Thank you for installing Anaconda Distribution.

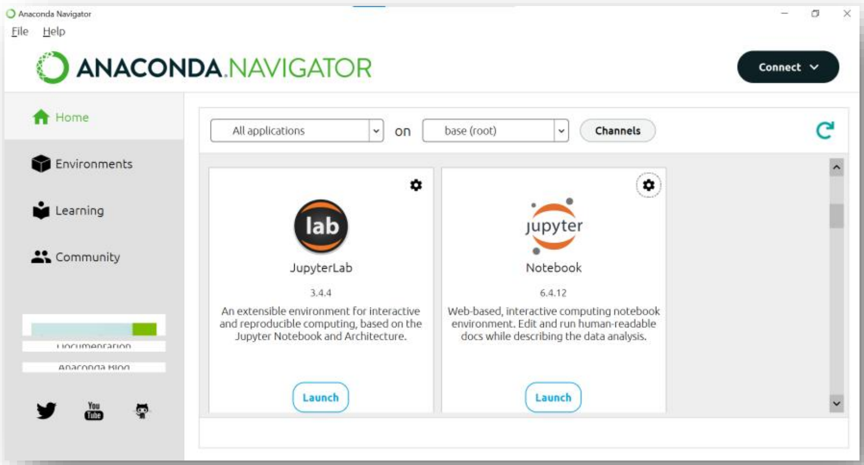
Here are some helpful tips and resources to get you started. We recommend you bookmark these links so you can refer back to them later.

☑ Anaconda Distribution Tutorial

☑ Getting Started with Anaconda

Now open Anaconda and carry on with the following tasks.

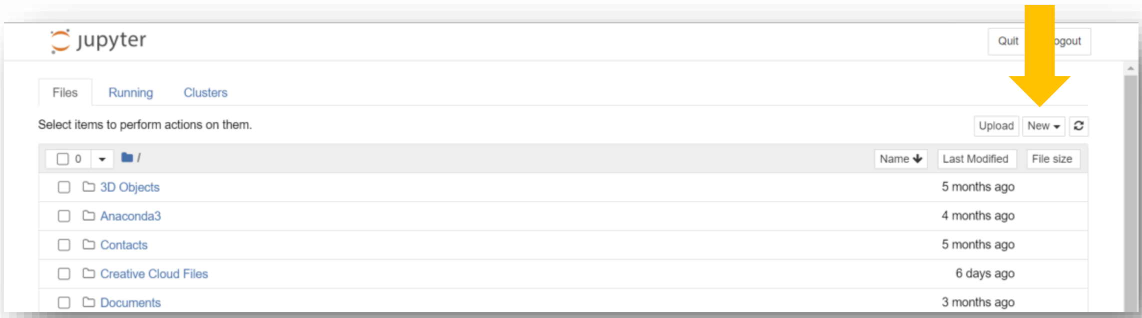
1- Open Anaconda and launch Jupyter Notebook



An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

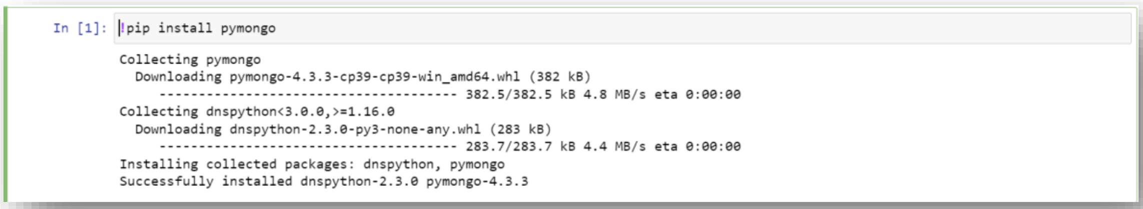
2- In the Jupyter Notebook create a new notebook.



Select items to perform actions on them.

Name	Last Modified	File size
3D Objects	5 months ago	
Anaconda3	4 months ago	
Contacts	5 months ago	
Creative Cloud Files	6 days ago	
Documents	3 months ago	

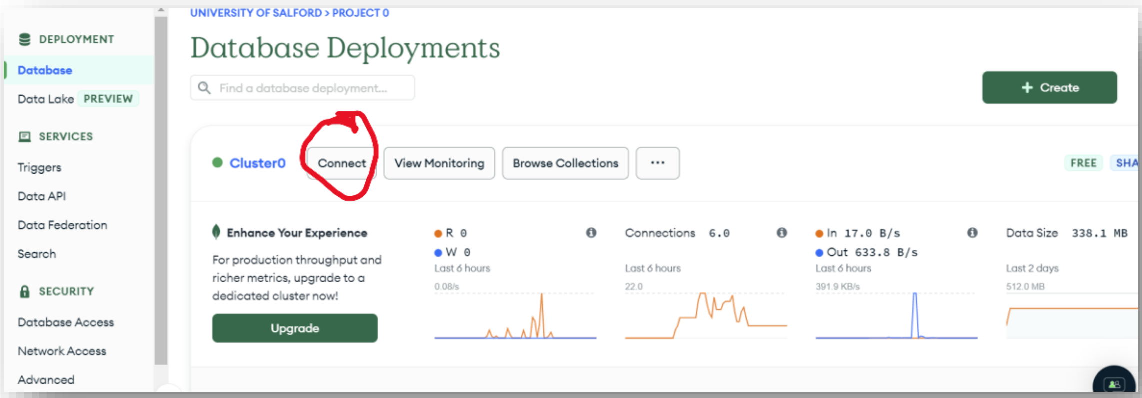
3- pymongo is the required library to work with MongoDB in Python. Install it on your Jupyter notebook. You need to do this just once.



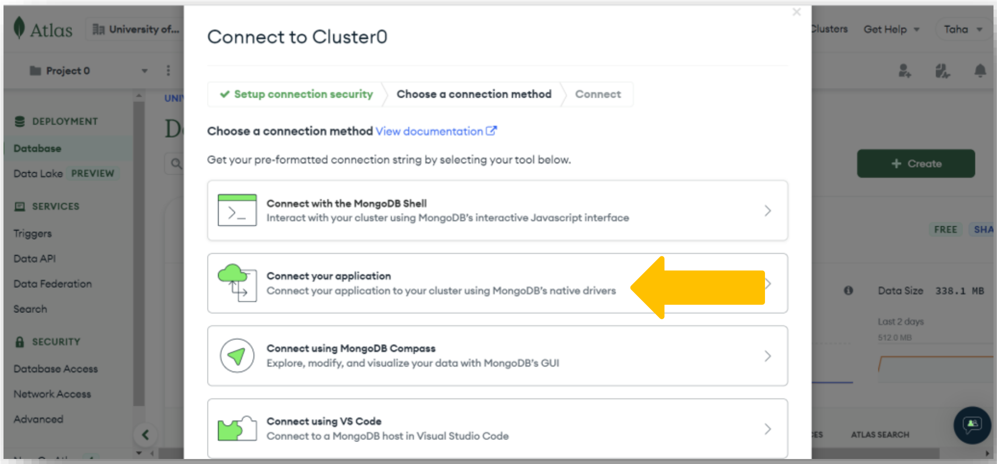
```
In [1]: !pip install pymongo

        Collecting pymongo
          Downloading pymongo-4.3.3-cp39-cp39-win_amd64.whl (382 kB)
             ------------------------------------- 382.5/382.5 kB 4.8 MB/s eta 0:00:00
        Collecting dnspython<3.0.0,>=1.16.0
          Downloading dnspython-2.3.0-py3-none-any.whl (283 kB)
             ------------------------------------- 283.7/283.7 kB 4.4 MB/s eta 0:00:00
        Installing collected packages: dnspython, pymongo
        Successfully installed dnspython-2.3.0 pymongo-4.3.3
```

Note: MongoClient object is a part of pymongo. You need to pass a url containing most of the information required to access to MongoDB Atlas to instantiate from this Object. The url is your connection string. So first you should collect it from Atlas.

4- Go to Atlas, select your database and click on connect button.



5- Choose “connect your application” as your connection method.



Connect to Cluster0

✓ Setup connection security | Choose a connection method | Connect

Choose a connection method View documentation

Get your pre-formatted connection string by selecting your tool below.

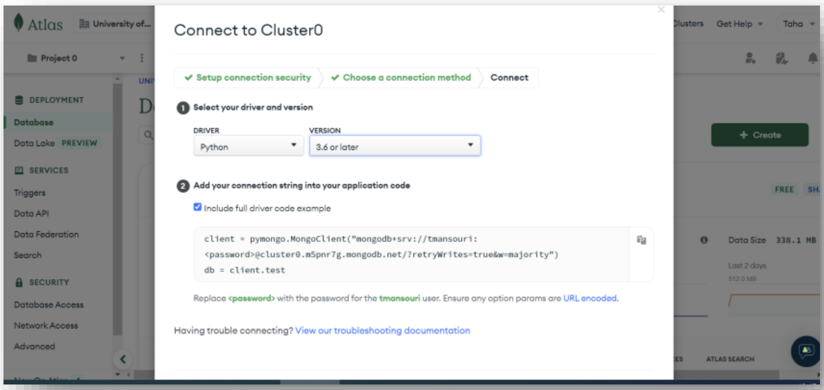
Connect with the MongoDB Shell
Interact with your cluster using MongoDB's interactive Javascript interface

Connect your application
Connect your application to your cluster using MongoDB's native drivers

Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI

Connect using VS Code
Connect to a MongoDB host in Visual Studio Code

6- Select Python and version later than 3.6. Then copy the generated connection string. You have to replace your own password and check “include full driver code example.



Connect to Cluster0

✓ Setup connection security ✓ Choose a connection method Connect

1 Select your driver and version

DRIVER: Python

VERSION: 3.6 or later

2 Add your connection string into your application code

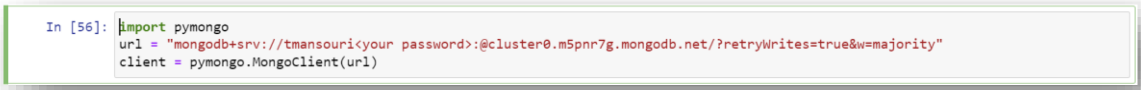
☑ Include full driver code example

```
client = pymongo.MongoClient("mongodb+srv://tmansouri:
<password>@cluster0.m5pnr7g.mongodb.net/?retryWrites=true&w=majority")
db = client.test
```

Replace <password> with the password for the tmansouri user. Ensure any option params are URL encoded.

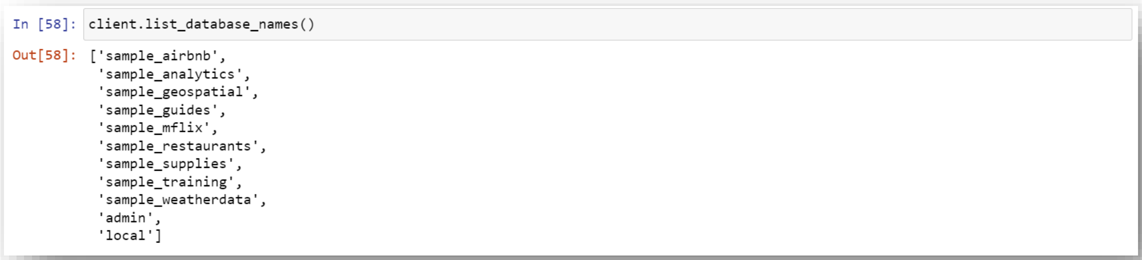
Having trouble connecting? View our troubleshooting documentation

7- Go back to Jupyter Notebook, import pymongo, define your url based on step 6 and instantiate a client.



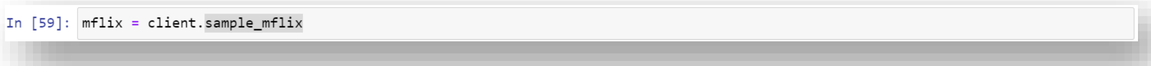
```
In [56]: import pymongo
         url = "mongodb+srv://tmansouri<your password>:@cluster0.m5pnr7g.mongodb.net/?retryWrites=true&w=majority"
         client = pymongo.MongoClient(url)
```

8- We can list the databases connected to this client object through the following command:



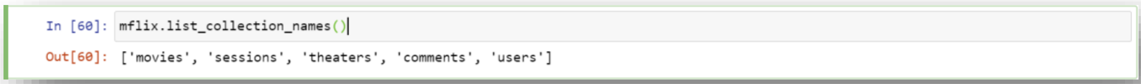
```
In [58]: client.list_database_names()

Out[58]: ['sample_airbnb',
          'sample_analytics',
          'sample_geospatial',
          'sample_guides',
          'sample_mflix',
          'sample_restaurants',
          'sample_supplies',
          'sample_training',
          'sample_weatherdata',
          'admin',
          'local']
```

9- Select sample_mflix database.



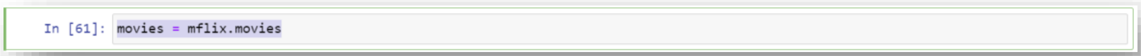
```
In [59]: mflix = client.sample_mflix
```

10- Now list its collections.



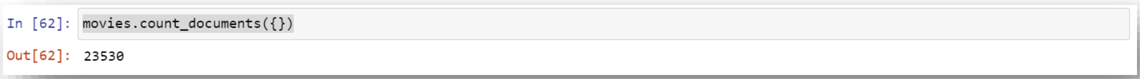
```
In [60]: mflix.list_collection_names()

Out[60]: ['movies', 'sessions', 'theaters', 'comments', 'users']
```

11- Select movies collection and define a new object upon that collection.



```
In [61]: movies = mflix.movies
```

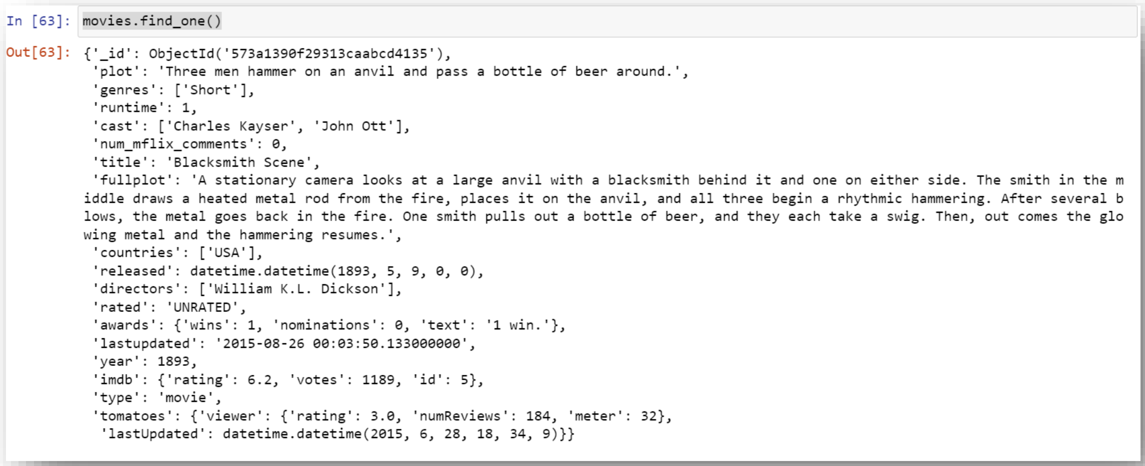
12- Count the number of documents in this collection.



```
In [62]: movies.count_documents({})

Out[62]: 23530
```

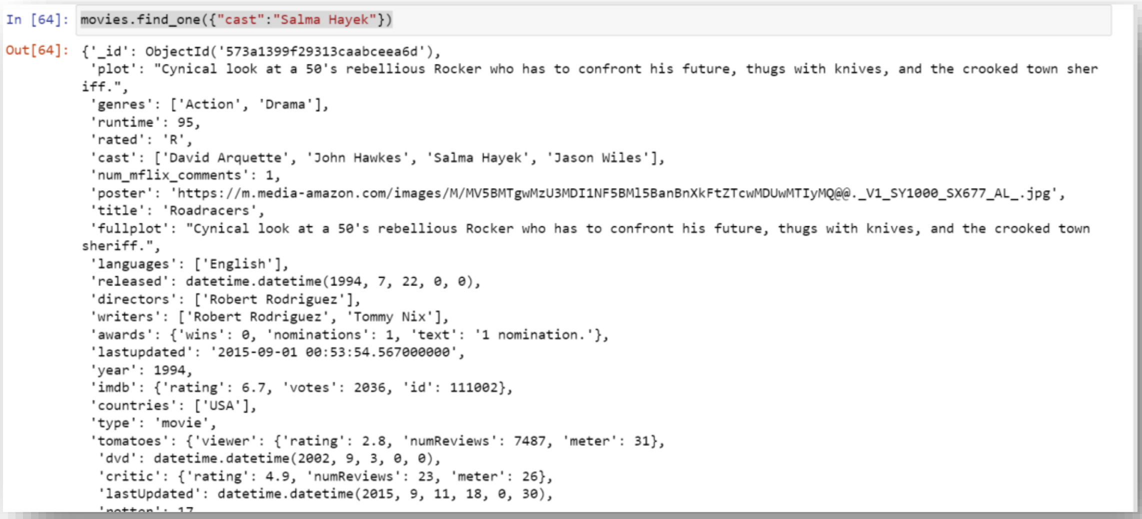
Note: there are two methods to read from a collection. find_one() that returns the first document satisfying the defined condition(s) in a natural order.

13- Find a movie.



```
In [63]: movies.find_one()

Out[63]: {'_id': ObjectId('573a1390f29313caabcd4135'),
          'plot': 'Three men hammer on an anvil and pass a bottle of beer around.',
          'genres': ['Short'],
          'runtime': 1,
          'cast': ['Charles Kayser', 'John Ott'],
          'num_mflix_comments': 0,
          'title': 'Blacksmith Scene',
          'fullplot': 'A stationary camera looks at a large anvil with a blacksmith behind it and one on either side. The smith in the m
         iddle draws a heated metal rod from the fire, places it on the anvil, and all three begin a rhythmic hammering. After several b
         lows, the metal goes back in the fire. One smith pulls out a bottle of beer, and they each take a swig. Then, out comes the glo
         wing metal and the hammering resumes.',
          'countries': ['USA'],
          'released': datetime.datetime(1893, 5, 9, 0, 0),
          'directors': ['William K.L. Dickson'],
          'rated': 'UNRATED',
          'awards': {'wins': 1, 'nominations': 0, 'text': '1 win.'},
          'lastupdated': '2015-08-26 00:03:50.133000000',
          'year': 1893,
          'imdb': {'rating': 6.2, 'votes': 1189, 'id': 5},
          'type': 'movie',
          'tomatoes': {'viewer': {'rating': 3.0, 'numReviews': 184, 'meter': 32},
           'lastUpdated': datetime.datetime(2015, 6, 28, 18, 34, 9)}}
```

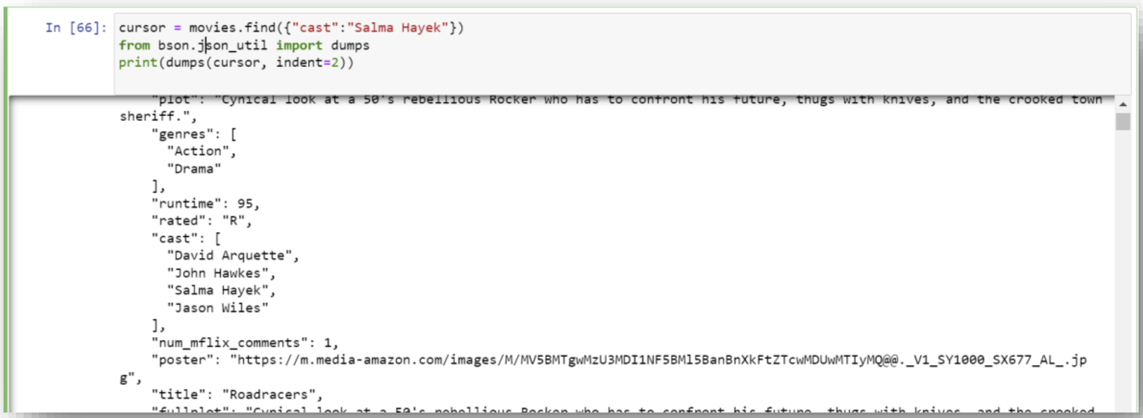
14- Find a movie casted by Salma Hayek. So, you need to pass a dictionary to find_one method containing a field name and the associated condition.



```
In [64]: movies.find_one({"cast":"Salma Hayek"})

Out[64]: {'_id': ObjectId('573a1399f29313caabceea6d'),
          'plot': "Cynical look at a 50's rebellious Rocker who has to confront his future, thugs with knives, and the crooked town sher
         iff.",
          'genres': ['Action', 'Drama'],
          'runtime': 95,
          'rated': 'R',
          'cast': ['David Arquette', 'John Hawkes', 'Salma Hayek', 'Jason Wiles'],
          'num_mflix_comments': 1,
          'poster': 'https://m.media-amazon.com/images/M/MV5BMTgwMzU3MDI1NF5BMl5BanBnXkFtZTcwMDUwMTIyMQ@@._V1_SY1000_SX677_AL_.jpg',
          'title': 'Roadracers',
          'fullplot': "Cynical look at a 50's rebellious Rocker who has to confront his future, thugs with knives, and the crooked town
         sheriff.",
          'languages': ['English'],
          'released': datetime.datetime(1994, 7, 22, 0, 0),
          'directors': ['Robert Rodriguez'],
          'writers': ['Robert Rodriguez', 'Tommy Nix'],
          'awards': {'wins': 0, 'nominations': 1, 'text': '1 nomination.'},
          'lastupdated': '2015-09-01 00:53:54.567000000',
          'year': 1994,
          'imdb': {'rating': 6.7, 'votes': 2036, 'id': 111002},
          'countries': ['USA'],
          'type': 'movie',
          'tomatoes': {'viewer': {'rating': 2.8, 'numReviews': 7487, 'meter': 31},
           'dvd': datetime.datetime(2002, 9, 3, 0, 0),
           'critic': {'rating': 4.9, 'numReviews': 23, 'meter': 26},
           'lastUpdated': datetime.datetime(2015, 9, 11, 18, 0, 30),
```

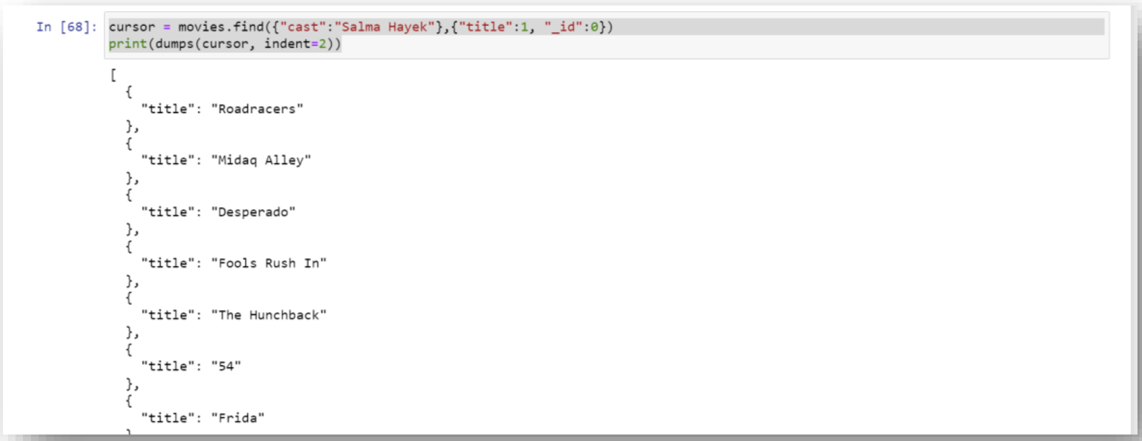
Note: Most of the time, find_one is not what we want to use, as we typically want to find all documents satisfying a condition. On this occasion we use find() method. Find() doesn't return a response – instead it returns a cursor object. We can store the cursor in a variable and dump it from a JSON format. Now we can access the documents inside of the cursor. Dump is in the JSON library and gives us an output in a nice format.

15- Find all movies with Salma Hayek in the cast and print them.



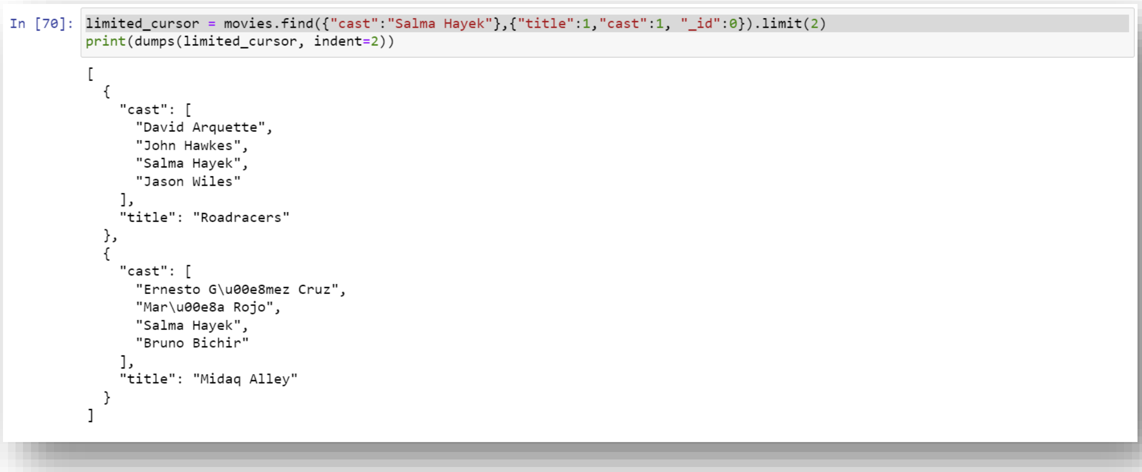
```
In [66]: cursor = movies.find({"cast":"Salma Hayek"})
         from bson.json_util import dumps
         print(dumps(cursor, indent=2))

         sheriff.",
             "genres": [
               "Action",
               "Drama"
             ],
             "runtime": 95,
             "rated": "R",
             "cast": [
               "David Arquette",
               "John Hawkes",
               "Salma Hayek",
               "Jason Wiles"
             ],
             "num_mflix_comments": 1,
             "poster": "https://m.media-amazon.com/images/M/MV5BMTgwMzU3MDI1NF5BMl5BanBnXkFtZTcwMDUwMTIyMQ@@._V1_SY1000_SX677_AL_.jp
         g",
             "title": "Roadracers",
```

16- Now what happens once you don't need to have all the fields? In this situation you need to specify the projection list. The second dictionary is a projection list, containing the field name and either a 1 (meaning that you want to show it) or a 0 (meaning you want to omit it). Find movies casted by Salma Hayek and just print their title.



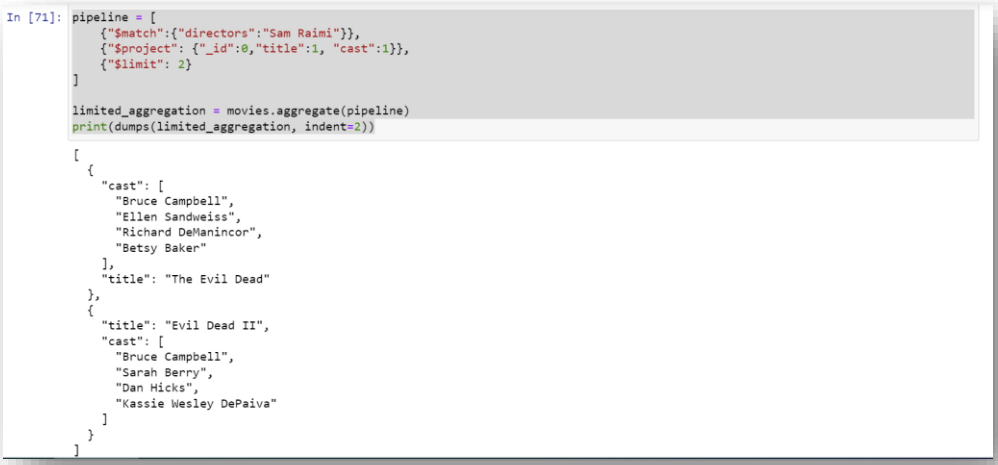
```
In [68]: cursor = movies.find({"cast":"Salma Hayek"},{"title":1, "_id":0})
         print(dumps(cursor, indent=2))

         [
           {
             "title": "Roadracers"
           },
           {
             "title": "Midaq Alley"
           },
           {
             "title": "Desperado"
           },
           {
             "title": "Fools Rush In"
           },
           {
             "title": "The Hunchback"
           },
           {
             "title": "54"
           },
           {
             "title": "Frida"
```

17- You can also limit the number of documents returned by pymonogo through limit(). Just show two documents regarding the above conditions.



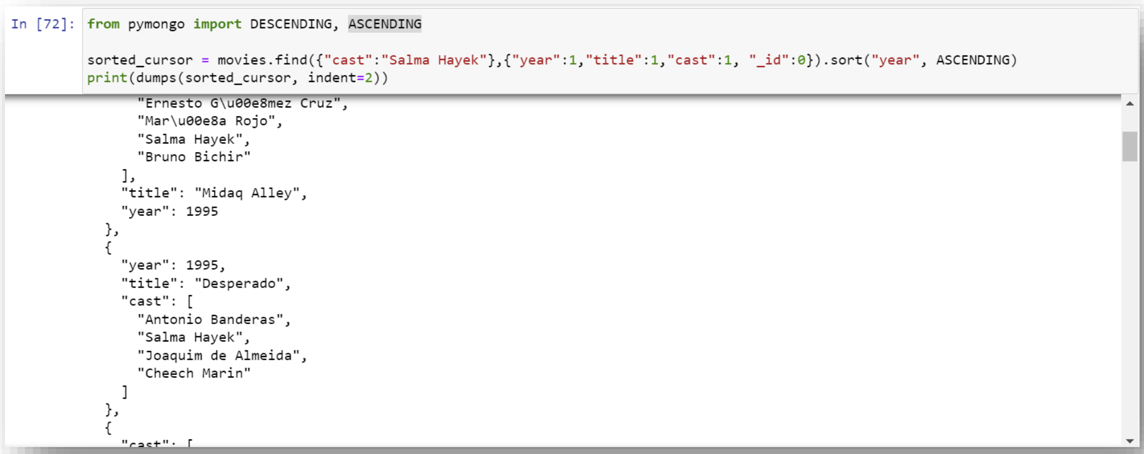
```
In [70]: limited_cursor = movies.find({"cast":"Salma Hayek"},{"title":1,"cast":1, "_id":0}).limit(2)
         print(dumps(limited_cursor, indent=2))

         [
           {
             "cast": [
               "David Arquette",
               "John Hawkes",
               "Salma Hayek",
               "Jason Wiles"
             ],
             "title": "Roadracers"
           },
           {
             "cast": [
               "Ernesto G\u00e8mez Cruz",
               "Mar\u00e8a Rojo",
               "Salma Hayek",
               "Bruno Bichir"
             ],
             "title": "Midaq Alley"
           }
         ]
```

18- You can design the above command through the aggregation framework.



```
In [71]: pipeline = [
             {"$match":{"directors":"Sam Raimi"}},
             {"$project": {"_id":0,"title":1, "cast":1}},
             {"$limit": 2}
         ]

         limited_aggregation = movies.aggregate(pipeline)
         print(dumps(limited_aggregation, indent=2))

         [
           {
             "cast": [
               "Bruce Campbell",
               "Ellen Sandweiss",
               "Richard DeManincor",
               "Betsy Baker"
             ],
             "title": "The Evil Dead"
           },
           {
             "title": "Evil Dead II",
             "cast": [
               "Bruce Campbell",
               "Sarah Berry",
               "Dan Hicks",
               "Kassie Wesley DePaiva"
             ]
           }
         ]
```

19- The next operator is sorting. Sort() method takes two parameter including key and the sorting order “ASCENDING” or “DESCENDING”. Sort movies casted by Salma Hayek based on their production year in the ascending order.



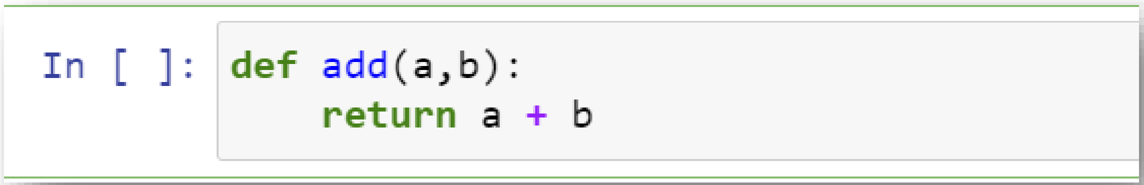
```
In [72]: from pymongo import DESCENDING, ASCENDING

         sorted_cursor = movies.find({"cast":"Salma Hayek"},{"year":1,"title":1,"cast":1, "_id":0}).sort("year", ASCENDING)
         print(dumps(sorted_cursor, indent=2))
```

```
      "Ernesto G\u00e8mez Cruz",
      "Mar\u00e8a Rojo",
      "Salma Hayek",
      "Bruno Bichir"
    ],
    "title": "Midaq Alley",
    "year": 1995
  },
  {
    "year": 1995,
    "title": "Desperado",
    "cast": [
      "Antonio Banderas",
      "Salma Hayek",
      "Joaquim de Almeida",
      "Cheech Marin"
    ]
  },
  {
    "cast": [
```

Note: Aggregation is a pipeline. Pipelines are composed of stages, which are broad units of work. Within stages, expressions are used to specify individual units of works. Expressions are functions. Each stage is like an assembly station and does a specific task. For example, $match checks a specific condition and select documents that fulfil that condition, $projection filters out unnecessary fields, and $group collects them together.

Let’s look at a function called add in Python and Aggregation framework.

Python

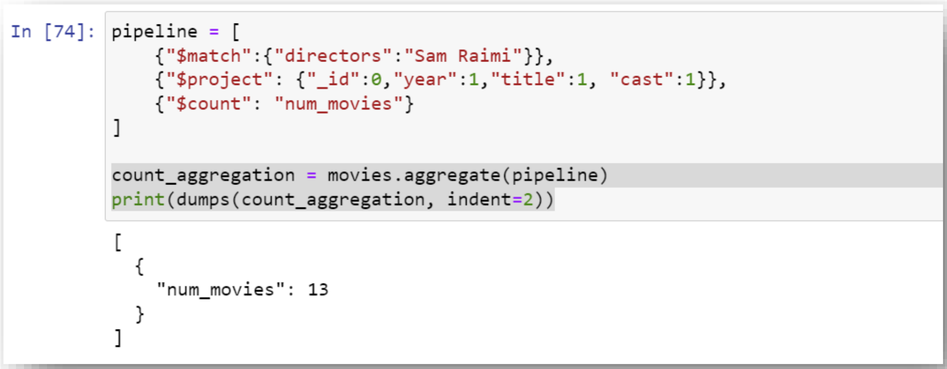


```
In [ ]: def add(a,b):
            return a + b
```

Aggregation Framework

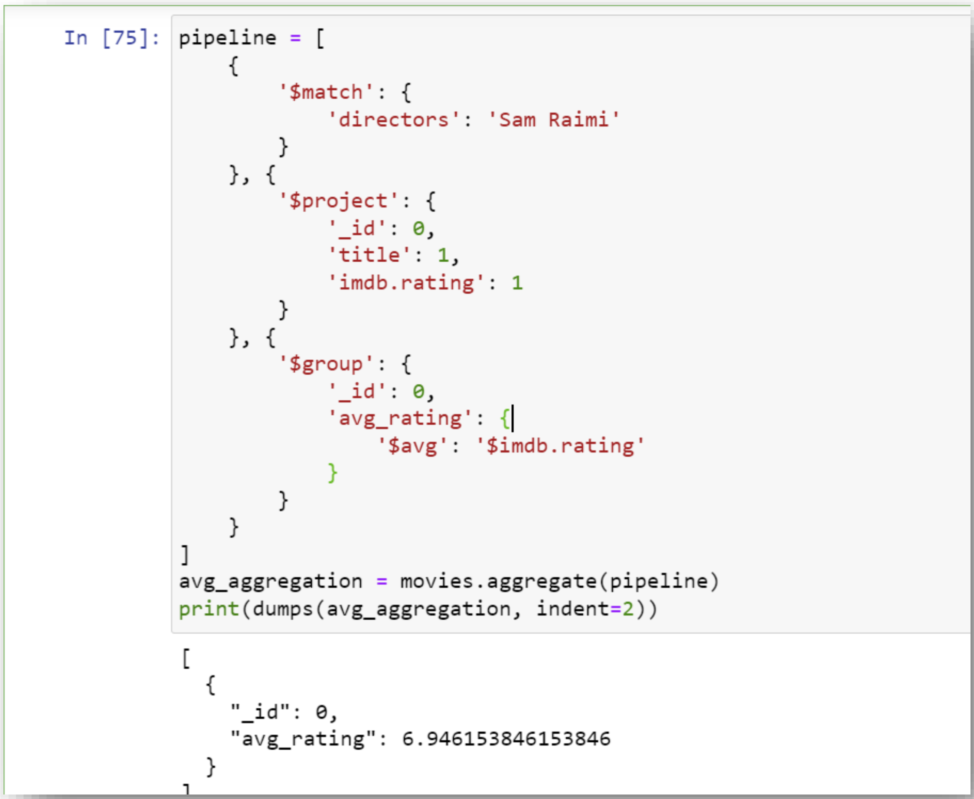
{“$add” :[“$a”, “$b”]}; all stages in the aggregation framework have $ before them.

20- Count the number of movies directed by Sam Raimi



```
In [74]: pipeline = [
             {"$match":{"directors":"Sam Raimi"}},
             {"$project": {"_id":0,"year":1,"title":1, "cast":1}},
             {"$count": "num_movies"}
         ]

         count_aggregation = movies.aggregate(pipeline)
         print(dumps(count_aggregation, indent=2))

         [
           {
             "num_movies": 13
           }
         ]
```

21- Run the exported pipeline of the first aggregation in Part-4 in Python.



```
In [75]: pipeline = [
             {
                 '$match': {
                     'directors': 'Sam Raimi'
                 }
             }, {
                 '$project': {
                     '_id': 0,
                     'title': 1,
                     'imdb.rating': 1
                 }
             }, {
                 '$group': {
                     '_id': 0,
                     'avg_rating': {
                         '$avg': '$imdb.rating'
                     }
                 }
             }
         ]
         avg_aggregation = movies.aggregate(pipeline)
         print(dumps(avg_aggregation, indent=2))

         [
           {
             "_id": 0,
             "avg_rating": 6.946153846153846
           }
```

Challenge 2

Implement the pipeline designed for challenge-2 in Python.

References:

- MongoDB University. MongoDB offers a range of online courses and certifications that cover various aspects of MongoDB development, administration, and deployment. These courses are self-paced and are designed to help you learn at your own pace (https://learn.mongodb.com/).
- https://www.mongodb.com/docs/manual/tutorial/query-documents/