# Working with Big Data in Python

**Section 1: Working with MongoDB**

**1.1 The Course Overview**

This video provides an overview of the entire course.

**1.2 What Is MongoDB and Why Should I Use It?**

Let us explore and discover what a non-relational database is. MongoDB is a non-relational database.

* Review different choices of formats to store data
* Compare structured and unstructured data
* Discover the strengths of MongoDB to build data pipelines

**1.3 From Tabular Data to JSON Documents**

Often, we consider data in tabular form, but JSON data is the data format of modern web applications. JavaScript Object Notation is a natural format for big data applications.

* Learn how and why to manipulate JSON data
* Learn how to store data on personnel in tabular and nested formats: a comparison
* Learn how to map common SQL operations and concepts to MongoDB

**1.4 MongoDB Indices and Datatypes**

MongoDB, like SQL can use indexed data to speed up common queries.

* Learn how indexing can speed up queries
* Understand how Port SQL SELECT queries to MongoDB
* Learn the different datatypes supported by MongoDB

**1.5 Setting Up MongoDB and Running Our First MongoDB Query**

Get up and running with fundamental MongoDB operations like creating a database and storing/retrieving documents through the Mongo shell.

* Install and start running the mongo daemon
* Create a MongoDB database and collection
* Insert and retrieve documents

**Section 2: Using the pyMongo Module**

**2.1 Setting Up pyMongo**

pyMongo is the Python API for MongoDB. Let us learn how to get up and running with pyMongo.

* Install the Python Anaconda distribution for data science
* Start up your MongoDB server
* Execute basic MongoDB commands through pyMongo

**2.2 Using pyMongo Cursors**

Often our queries return large numbers of documents through cursors. Let us learn how cursors work, so we can deal with these queries without a large memory footprint.

* Review iterable objects in Python
* Review generator objects in Python
* Good practice in opening and closing connections with cursors

**2.3 Inserting and Finding Documents**

MongoDB provides a rich syntax to precisely control which data is returned from a query using the query and projection operators. Learn how to specify these arguments.

* Control and filter the documents using query
* Control the fields which are returned from the matching documents
* Use dot notation to make queries as Embedded documents.

**2.4 Return Codes and Exceptions**

MongoDB is a highly scalable database capable of many simultaneous connections. Sometimes this causes errors in operations. Learn how to deal with these errors.

* Concurrency requires locks and queues of operations
* try/except clauses to catch errors
* Specific error handling using pymongo.errors

**2.5 Using Operators, Updates, and Aggregations**

Using what you have learnt about finding and matching documents also learn how to use operators and the aggregate function to calculate aggregated statistics and update documents.

* Update documents using update operators
* Review comparison operators
* Calculate statistics using aggregate()

**Section 3: Example 1: Loading and Querying Weather Data**

**3.1 Grabbing Weather Data via OpenWeather API**

Web APIs are a common source of data, learn how to leverage pyMongo and requests to extract useful information from API data.

* Use the requests library to define and execute an HTTP request
* Throttle requests to respect API usage limits
* Scale requests up through abstraction

**3.2 Ingesting Weather Data into MongoDB**

Store JSON data returned from the Open Weather API to be later queried.

* Define a collection to store weather data
* Loop through results and catch any MongoDB errors on insertion
* Sanity check inserted documents in MongoDB

**3.3 Querying Weather Data from MongoDB**

Raw data doesn’t provide insights; develop pipeline aggregation operations to summarise and filter data in an iterative fashion.

* Filter out noisy data resulting from unexpected behavior
* Group weather data by city
* Extract maximum temperature information

**Section 4: Working with pySpark and MongoDB**

**4.1 What Is Spark and When Do We Need It?**

Modern datasets are challenging to process as our memory and processing needs are large and variable. Spark helps to scale analysis over a cluster of processors.

* Understand the concepts underlying Spark
* Learn the origins of Spark
* Install pySpark on your own machine

**4.2 Data Structures in Spark**

Spark data structures are key to building effective processing pipelines; understand the difference between RDDs and dataframes.

* Explore the fundamental structure - RDDs in Spark and learn how to store unstructured data
* Use the Spark machine learning library - spark.ml and use dataframes to store the structured data
* Learn how to create dataframes and RDDs from text and JSON data files

**4.3 Data Structures in Spark (Continued)**

Spark can be a hard framework to get started. Implement the data structures and ideas from the previous video in pySpark code.

* Create an RDD from a text file
* Parse lines into a usable RDD through transformations
* Create a dataframe from an RDD

**4.4 Connecting to MongoDB with PySpark**

If our data resides in MongoDB, we need to extract it into a Spark data structure to analyze it.

* Learn how to use the pySpark mongo connector and understand Spark packages
* Invoke pySpark with the necessary library to connect to MongoDB
* Ingest the results of a mongoDB query into a Spark dataframe

**Section 5: Example 2: Querying Reddit Comment Data with MongoDB and PySpark**

**5.1 Making Reddit Data Available to PySpark**

JSON data often exists in data dumps rather than being extracted from an API incrementally. Reddit is a popular site for posting and commenting.

* Learn how to do bulk imports into MongoDB
* Locate Reddit data from open data dumps by month and year
* Use mongoimport to directly ingest a JSON file into a collection

**5.2 Loading Data from MongoDB in Spark, Transform into Pandas DF**

Data stored in MongoDB needs to be made available to Spark in a data structure that is valid.

* Use spark.read.format() to read from MongoDB into a spark dataframe
* Pass pipeline options to spark.read.format()
* Derive length of post word length

**5.3 Preparing Data for Prediction Task Using spark.ml**

Spark.ml requires dataframe columns to be of type Vector. Spark dataframe columns can store data of many different types.

* Use VectorAssembler to transform into the correct format
* Transform and clean data using Spark functions
* Use VectorAssember to prepare multiple dataframe columns into a single features column for spark.ml

**5.4 Predicting Up Votes Using pyspark.ml**

The popularity of Reddit comments depends on lots of aspects, can a predictive model help understand how to have popular posts? Let us explore and find an answer to this in this video.

* Use a linear regression model to predict popularity
* Train and test a linear regression model
* Consider alternative features to improve predictions of popularity