These are the links in this lecture:

* + [City of San Francisco online data](https://data.sfgov.org/)
  + [City of San Francisco 3-1-1 report data](http://www.sf311.org/)
  + [Structured Open Urban Data: Understanding the Landscape](http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4174913/pdf/big.2014.0020.pdf) paper
  + Spark [online documentation](https://spark.apache.org/docs/latest/sql-programming-guide.html) for DataFrames
  + Spark [DataFrame API](https://spark.apache.org/docs/latest/api/python/pyspark.sql.html" \l "pyspark.sql.DataFrame" \t "_blank)
  + [Apache Common Log Format specification](http://httpd.apache.org/docs/2.4/logs.html#common)
  + Two month's worth of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida in 1995 are available online [here](http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html).
  + Wikipedia references:
    - [Text files](https://en.wikipedia.org/wiki/Text_file)
    - [ASCII](https://en.wikipedia.org/wiki/ASCII)
    - [Unicode](https://en.wikipedia.org/wiki/Unicode)
    - [Entropy](https://en.wikipedia.org/wiki/Entropy_(information_theory))
    - [Binary to text encoding](https://en.wikipedia.org/wiki/Binary-to-text_encoding)
    - [Binary files](https://en.wikipedia.org/wiki/Binary_file)
    - [Compression](https://en.wikipedia.org/wiki/Data_compression)
    - [Lossy compression](https://en.wikipedia.org/wiki/Lossy_data_compression)
    - [Lossless compression](https://en.wikipedia.org/wiki/Lossless_data_compression)
    - [JPEG](https://en.wikipedia.org/wiki/JPEG)
    - [MP3](https://en.wikipedia.org/wiki/MP3)
    - [MPEG-4](https://en.wikipedia.org/wiki/MPEG-4)
    - [Gzip](https://en.wikipedia.org/wiki/Gzip)
    - [LZ4](https://en.wikipedia.org/wiki/LZ4_(compression_algorithm))
    - [PNG](https://en.wikipedia.org/wiki/Portable_Network_Graphics)
    - [Dolby TrueHD](https://en.wikipedia.org/wiki/Dolby_TrueHD)

Overview

The City of San Francisco has an extensive collection of [online city records](https://data.sfgov.org/). These data cover public safety, health, transportation, housing and many other topics. The data, together with public social media, can provide an unprecedented window into the City's operations. The data ia is freely available for anyone to explore. Many other cities are also putting their records online. Let's consider a couple of the types of  questions one can ask using this data, but there are many others:

Questions

1. In the health section of [sf.data.gov](https://data.sfgov.org/) there is an extensive set of records about restaurant inspections. San Francisco has the largest number of restaurants per capita of any major city in the United States. Tracking and maintaining the quality of those restaurants is an ongoing challenge for inspectors. An interesting question would be could you create an "early warning system" based on social media (e.g., is it possible to predict restaurants in need of inspection from Yelp reviews?)? This question could be partially answered by building a machine learning classifier using historical social media reviews (e.g., from Yelp or Trip Advisor) and city records of inspections.
2. The City recieves many [3-1-1 reports](http://www.sf311.org/) (non-emergency incident reports from citizens). But some of these reports predict serious incidents, which may be recorded later in police reports. Consider the challenge of mining the CABLE reported incidents, and looking for text markers that predict future police reports. This challenge would require tieing the two tabular datasets together, a process that is complicated by noisy data - would it be better to tie the datasets together by the name of the protagonist or the location (address) of the incident? This is an entity resolution problem, as the protagonists' names might be listed differently in the two datasets (e.g., Anthony Joseph versus A. Joseph), and the same applies to address information in the two datasets (e.g., SF City Hall versus 1 Dr Carlton B Goodlett Pl, San Francisco, CA 94102). You will explore the entity resolution problem in Lab #3.  After combining the two datasets, the next step would be to look for keywords in the police report marking the type of incident, and attempt to predict incidents from the full text of the CABLE report.

These two are just two challenges, but there are many more are possible ones from this dataset.

References

(Optional Reading) This paper, [Structured Open Urban Data: Understanding the Landscape](http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4174913/pdf/big.2014.0020.pdf), examines over 9,000 open data sets from 20 cities in North America, and presents general statistics about the content, size, nature, and popularity of the different data sets, and also examines the data quality issues and time-related aspects of the various datasets.

# Spark DataFrame References

You can find more information about Spark DataFrames in the Spark [online documentation](https://spark.apache.org/docs/latest/sql-programming-guide.html) for DataFrames. The [DataFrame API](https://spark.apache.org/docs/latest/api/python/pyspark.sql.html" \l "pyspark.sql.DataFrame" \t "_blank) is also available online.

# Apache Common Log Format

You can find the specification for the Apache CLR discussed in this segment online [here](http://httpd.apache.org/docs/2.4/logs.html#common).

**NASA Web Server Log**

Two month's worth of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida in 1995 are available online [here](http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html).

## TEXT FILES

[Text files](https://en.wikipedia.org/wiki/Text_file) typically contain lines of [ASCII](https://en.wikipedia.org/wiki/ASCII) or [Unicode](https://en.wikipedia.org/wiki/Unicode) characters and may use a human-readable format. Because they are human-readable, text files usually have low [entropy](https://en.wikipedia.org/wiki/Entropy_(information_theory)) which means that they take up more space than necessary to represent the information they contain. Text files can contain binary data, but that data must be [encoded in textual form](https://en.wikipedia.org/wiki/Binary-to-text_encoding). Examples of text files are Comma Separated Value files,  Python files, and even iPython notebook files.

## BINARY FILES

[Binary files](https://en.wikipedia.org/wiki/Binary_file) are files that are not text files. They consist of sequences of bytes and may contain parts that can be interpreted as text, and may contain headers, or blocks of metadata that can used by a computer program to interpret the data in the file. Examples of binary files are images, video and sound files, and computer programs.

## COMPRESSION

[Compression](https://en.wikipedia.org/wiki/Data_compression) is used to store a file's contents in a binary format that more information dense, and thus typically takes up less space than the original contents. There are two types of compression:[lossy](https://en.wikipedia.org/wiki/Lossy_data_compression" \t "_blank) and [lossless](https://en.wikipedia.org/wiki/Lossless_data_compression). When compression is applied to a text file, the resulting information must be stored in a binary file.

Lossy compression by its name, means compression where information is lost or discarded. Examples are image, sound, and video compression. There a various lossy compression techniques that take advantage of the way that we see or hear content to discard extra information without being our noticing the differences (in most cases) between the original and the compressed content. Examples of lossy compression are [JPEG](https://en.wikipedia.org/wiki/JPEG) for images, [MP3](https://en.wikipedia.org/wiki/MP3) for sound files, and [MPEG-4](https://en.wikipedia.org/wiki/MPEG-4) for video files.

Lossless compression is a set of techniques that reduces the amount of space that data takes without discarding any of the data, and it is used in cases where it is important that the original and the decompressed data be identical. A lossless compression algorithm works in two steps:  first, it generates a statistical model for the input data, and second, it uses this model to map input data to bit sequences in such a way that "probable" (e.g., frequently encountered) data will produce shorter output than "improbable" data. Two examples of lossless compression algorithms are [Gzip](https://en.wikipedia.org/wiki/Gzip" \t "_blank) and [LZ4](https://en.wikipedia.org/wiki/LZ4_(compression_algorithm)). Each of these algorithms has tradeoffs as discussed in the lecture segment video. Examples of lossless compression are [PNG](https://en.wikipedia.org/wiki/Portable_Network_Graphics) for images and [Dolby TrueHD](https://en.wikipedia.org/wiki/Dolby_TrueHD) for sound files.