**Data Pipelines: ETL vs ELT**

Data pipeline is a generic term for moving data from one place to another. For example, it could be moving data from one server to another server.

**ETL**

An [ETL pipeline](https://en.wikipedia.org/wiki/Extract,_transform,_load) is a specific kind of data pipeline and very common. ETL stands for Extract, Transform, Load. Imagine that you have a database containing web [log data](https://en.wikipedia.org/wiki/Log_file). Each entry contains the IP address of a user, a timestamp, and the link that the user clicked.

What if your company wanted to run an analysis of links clicked by city and by day? You would need another data set that maps IP address to a city, and you would also need to extract the day from the timestamp. With an ETL pipeline, you could run code once per day that would extract the previous day's log data, map IP address to city, aggregate link clicks by city, and then load these results into a new database. That way, a data analyst or scientist would have access to a table of log data by city and day. That is more convenient than always having to run the same complex data transformations on the raw web log data.

Before cloud computing, businesses stored their data on large, expensive, private servers. Running queries on large data sets, like raw web log data, could be expensive both economically and in terms of time. But data analysts might need to query a database multiple times even in the same day; hence, pre-aggregating the data with an ETL pipeline makes sense.

**ELT**

ELT (extract, load, transform) pipelines have gained traction since the advent of cloud computing. Cloud computing has lowered the cost of storing data and running queries on large, raw data sets. Many of these cloud services, like [Amazon Redshift](https://aws.amazon.com/redshift/), [Google BigQuery](https://cloud.google.com/bigquery/), or [IBM Db2](https://www.ibm.com/cloud/db2-warehouse-on-cloud) can be queried using SQL or a SQL-like language. With these tools, the data gets extracted, then loaded directly, and finally transformed at the end of the pipeline.

However, ETL pipelines are still used even with these cloud tools. Oftentimes, it still makes sense to run ETL pipelines and store data in a more readable or intuitive format. This can help data analysts and scientists work more efficiently as well as help an organization become more data driven.

**Summary of the data file types you'll work with**

**CSV files**

CSV stands for comma-separated values. These types of files separate values with a comma, and each entry is on a separate line. Oftentimes, the first entry will contain variable names. Here is an example of what CSV data looks like. This is an abbreviated version of the first three lines in the World Bank projects data csv file.

id,regionname,countryname,prodline,lendinginstr

P162228,Other,World;World,RE,Investment Project Financing

P163962,Africa,Democratic Republic of the Congo;Democratic Republic of the Congo,PE,Investment Project Financing

**JSON**

JSON is a file format with key/value pairs. It looks like a Python dictionary. The exact same CSV file represented in JSON could look like this:

[{"id":"P162228","regionname":"Other","countryname":"World;World","prodline":"RE","lendinginstr":"Investment Project Financing"},{"id":"P163962","regionname":"Africa","countryname":"Democratic Republic of the Congo;Democratic Republic of the Congo","prodline":"PE","lendinginstr":"Investment Project Financing"},{"id":"P167672","regionname":"South Asia","countryname":"People\'s Republic of Bangladesh;People\'s Republic of Bangladesh","prodline":"PE","lendinginstr":"Investment Project Financing"}]

Each line in the data is inside of a squiggly bracket {}. The variable names are the keys, and the variable values are the values.

There are other ways to organize JSON data, but the general rule is that JSON is organized into key/value pairs. For example, here is a different way to represent the same data using JSON:

{"id":{"0":"P162228","1":"P163962","2":"P167672"},"regionname":{"0":"Other","1":"Africa","2":"South Asia"},"countryname":{"0":"World;World","1":"Democratic Republic of the Congo;Democratic Republic of the Congo","2":"People\'s Republic of Bangladesh;People\'s Republic of Bangladesh"},"prodline":{"0":"RE","1":"PE","2":"PE"},"lendinginstr":{"0":"Investment Project Financing","1":"Investment Project Financing","2":"Investment Project Financing"}}

**XML**

Another data format is called XML (Extensible Markup Language). XML is very similar to HTML at least in terms of formatting. The main difference between the two is that HTML has pre-defined tags that are standardized. In XML, tags can be tailored to the data set. Here is what this same data would look like as XML.

<ENTRY>

<ID>P162228</ID>

<REGIONNAME>Other</REGIONNAME>

<COUNTRYNAME>World;World</COUNTRYNAME>

<PRODLINE>RE</PRODLINE>

<LENDINGINSTR>Investment Project Financing</LENDINGINSTR>

</ENTRY>

<ENTRY>

<ID>P163962</ID>

<REGIONNAME>Africa</REGIONNAME>

<COUNTRYNAME>Democratic Republic of the Congo;Democratic Republic of the Congo</COUNTRYNAME>

<PRODLINE>PE</PRODLINE>

<LENDINGINSTR>Investment Project Financing</LENDINGINSTR>

</ENTRY>

<ENTRY>

<ID>P167672</ID>

<REGIONNAME>South Asia</REGIONNAME>

<COUNTRYNAME>People's Republic of Bangladesh;People's Republic of Bangladesh</COUNTRYNAME>

<PRODLINE>PE</PRODLINE>

<LENDINGINSTR>Investment Project Financing</LENDINGINSTR>

</ENTRY>

XML is falling out of favor especially because JSON tends to be easier to navigate; however, you still might come across XML data. The World Bank API, for example, can return either XML data or JSON data. From a data perspective, the process for handling HTML and XML data is essentially the same.

**SQL databases**

SQL databases store data in tables using [primary and foreign keys](https://docs.microsoft.com/en-us/sql/relational-databases/tables/primary-and-foreign-key-constraints?view=sql-server-2017). In a SQL database, the same data would look like this:

| **id** | **regionname** | **countryname** | **prodline** | **lendinginstr** |
| --- | --- | --- | --- | --- |
| P162228 | Other | World;World | RE | Investment Project Financing |
| P163962 | Africa | Democratic Republic of the Congo;Democratic Republic of the Congo | PE | Investment Project Financing |
| P167672 | South Asia | People's Republic of Bangladesh;People's Republic of Bangladesh | PE | Investment Project Financing |

**Text Files**

This course won't go into much detail about text data. There are other Udacity courses, namely on natural language processing, that go into the details of processing text for machine learning.

Text data present their own issues. Whereas CSV, JSON, XML, and SQL data are organized with a clear structure, text is more ambiguous. For example, the World Bank project data country names are written like this

Democratic Republic of the Congo;Democratic Republic of the Congo

In the World Bank Indicator data sets, the [Democratic Republic of the Congo](https://classroom.udacity.com/nanodegrees/nd025-ent/parts/6bb3fcfa-d138-46af-95b2-86350151cfe5/modules/f7cc9cca-a37b-4d8a-ba92-2b4edb90618d/lessons/013cd87c-acef-4cb1-b043-c47dbc4ef371/concepts/(https:/data.worldbank.org/country/congo-dem-rep?view=chart) is represented by the abbreviation "Congo, Dem. Rep." You'll have to clean these country names to join the data sets together.

**Extracting Data from the Web**

In this lesson, you'll see how to extract data from the web using an APIs (Application Programming Interface). APIs generally provide data in either JSON or XML format.

Companies and organizations provide APIs so that programmers can access data in an official, safe way. APIs allow you to download, and sometimes even upload or modify, data from a web server without giving you direct access.