

# *Data science in a big data world*

---

## ***This chapter covers***

- Defining data science and big data
- Recognizing the different types of data
- Gaining insight into the data science process
- Introducing the fields of data science and big data
- Working through examples of Hadoop

*Big data* is a blanket term for any collection of data sets so large or complex that it becomes difficult to process them using traditional data management techniques such as, for example, the RDBMS (relational database management systems). The widely adopted RDBMS has long been regarded as a one-size-fits-all solution, but the demands of handling big data have shown otherwise. *Data science* involves using methods to analyze massive amounts of data and extract the knowledge it contains. You can think of the relationship between big data and data science as being like the relationship between crude oil and an oil refinery. Data science and big data evolved from statistics and traditional data management but are now considered to be distinct disciplines.

The characteristics of big data are often referred to as the three Vs:

- *Volume*—How much data is there?
- *Variety*—How diverse are different types of data?
- *Velocity*—At what speed is new data generated?

Often these characteristics are complemented with a fourth V, veracity: How accurate is the data? These four properties make big data different from the data found in traditional data management tools. Consequently, the challenges they bring can be felt in almost every aspect: data capture, curation, storage, search, sharing, transfer, and visualization. In addition, big data calls for specialized techniques to extract the insights.

Data science is an evolutionary extension of statistics capable of dealing with the massive amounts of data produced today. It adds methods from computer science to the repertoire of statistics. In a research note from Laney and Kart, *Emerging Role of the Data Scientist and the Art of Data Science*, the authors sifted through hundreds of job descriptions for data scientist, statistician, and BI (Business Intelligence) analyst to detect the differences between those titles. The main things that set a data scientist apart from a statistician are the ability to work with big data and experience in machine learning, computing, and algorithm building. Their tools tend to differ too, with data scientist job descriptions more frequently mentioning the ability to use Hadoop, Pig, Spark, R, Python, and Java, among others. Don't worry if you feel intimidated by this list; most of these will be gradually introduced in this book, though we'll focus on Python. Python is a great language for data science because it has many data science libraries available, and it's widely supported by specialized software. For instance, almost every popular NoSQL database has a Python-specific API. Because of these features and the ability to prototype quickly with Python while keeping acceptable performance, its influence is steadily growing in the data science world.

As the amount of data continues to grow and the need to leverage it becomes more important, every data scientist will come across big data projects throughout their career.

## **1.1 Benefits and uses of data science and big data**

Data science and big data are used almost everywhere in both commercial and non-commercial settings. The number of use cases is vast, and the examples we'll provide throughout this book only scratch the surface of the possibilities.

Commercial companies in almost every industry use data science and big data to gain insights into their customers, processes, staff, completion, and products. Many companies use data science to offer customers a better user experience, as well as to cross-sell, up-sell, and personalize their offerings. A good example of this is Google AdSense, which collects data from internet users so relevant commercial messages can be matched to the person browsing the internet. MaxPoint (<http://maxpoint.com/us>)

is another example of real-time personalized advertising. Human resource professionals use people analytics and text mining to screen candidates, monitor the mood of employees, and study informal networks among coworkers. People analytics is the central theme in the book *Moneyball: The Art of Winning an Unfair Game*. In the book (and movie) we saw that the traditional scouting process for American baseball was random, and replacing it with correlated signals changed everything. Relying on statistics allowed them to hire the right players and pit them against the opponents where they would have the biggest advantage. Financial institutions use data science to predict stock markets, determine the risk of lending money, and learn how to attract new clients for their services. At the time of writing this book, at least 50% of trades worldwide are performed automatically by machines based on algorithms developed by *quants*, as data scientists who work on trading algorithms are often called, with the help of big data and data science techniques.

Governmental organizations are also aware of data's value. Many governmental organizations not only rely on internal data scientists to discover valuable information, but also share their data with the public. You can use this data to gain insights or build data-driven applications. *Data.gov* is but one example; it's the home of the US Government's open data. A data scientist in a governmental organization gets to work on diverse projects such as detecting fraud and other criminal activity or optimizing project funding. A well-known example was provided by Edward Snowden, who leaked internal documents of the American National Security Agency and the British Government Communications Headquarters that show clearly how they used data science and big data to monitor millions of individuals. Those organizations collected 5 billion data records from widespread applications such as Google Maps, Angry Birds, email, and text messages, among many other data sources. Then they applied data science techniques to distill information.

Nongovernmental organizations (NGOs) are also no strangers to using data. They use it to raise money and defend their causes. The World Wildlife Fund (WWF), for instance, employs data scientists to increase the effectiveness of their fundraising efforts. Many data scientists devote part of their time to helping NGOs, because NGOs often lack the resources to collect data and employ data scientists. DataKind is one such data scientist group that devotes its time to the benefit of mankind.

Universities use data science in their research but also to enhance the study experience of their students. The rise of massive open online courses (MOOC) produces a lot of data, which allows universities to study how this type of learning can complement traditional classes. MOOCs are an invaluable asset if you want to become a data scientist and big data professional, so definitely look at a few of the better-known ones: Coursera, Udacity, and edX. The big data and data science landscape changes quickly, and MOOCs allow you to stay up to date by following courses from top universities. If you aren't acquainted with them yet, take time to do so now; you'll come to love them as we have.

## 1.2 Facets of data

In data science and big data you'll come across many different types of data, and each of them tends to require different tools and techniques. The main categories of data are these:

- Structured
- Unstructured
- Natural language
- Machine-generated
- Graph-based
- Audio, video, and images
- Streaming

Let's explore all these interesting data types.

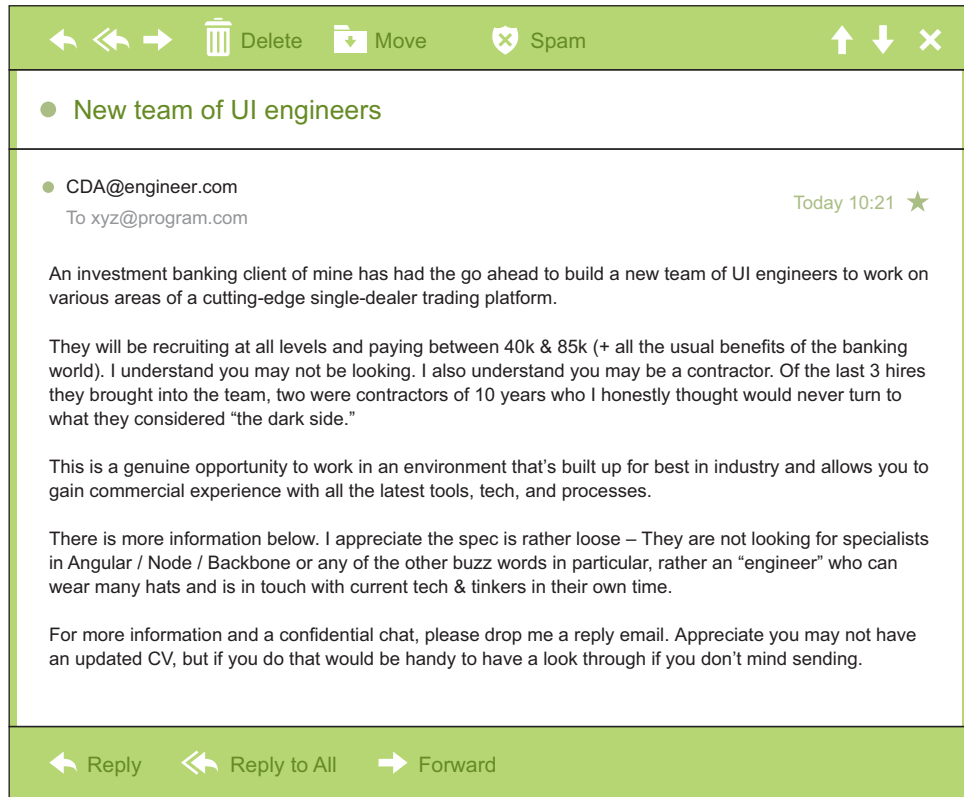
### 1.2.1 Structured data

Structured data is data that depends on a data model and resides in a fixed field within a record. As such, it's often easy to store structured data in tables within databases or Excel files (figure 1.1). SQL, or Structured Query Language, is the preferred way to manage and query data that resides in databases. You may also come across structured data that might give you a hard time storing it in a traditional relational database. Hierarchical data such as a family tree is one such example.

The world isn't made up of structured data, though; it's imposed upon it by humans and machines. More often, data comes unstructured.

1	Indicator ID	Dimension List	Timeframe	Numeric Value	Missing Value Flag	Confidence Int
2	214390830	Total (Age-adjusted)	2008	74.6%		73.8%
3	214390833	Aged 18-44 years	2008	59.4%		58.0%
4	214390831	Aged 18-24 years	2008	37.4%		34.6%
5	214390832	Aged 25-44 years	2008	66.9%		65.5%
6	214390836	Aged 45-64 years	2008	88.6%		87.7%
7	214390834	Aged 45-54 years	2008	86.3%		85.1%
8	214390835	Aged 55-64 years	2008	91.5%		90.4%
9	214390840	Aged 65 years and over	2008	94.6%		93.8%
10	214390837	Aged 65-74 years	2008	93.6%		92.4%
11	214390838	Aged 75-84 years	2008	95.6%		94.4%
12	214390839	Aged 85 years and over	2008	96.0%		94.0%
13	214390841	Male (Age-adjusted)	2008	72.2%		71.1%
14	214390842	Female (Age-adjusted)	2008	76.8%		75.9%
15	214390843	White only (Age-adjusted)	2008	73.8%		72.9%
16	214390844	Black or African American only (Age-adjusted)	2008	77.0%		75.0%
17	214390845	American Indian or Alaska Native only (Age-adjusted)	2008	66.5%		57.1%
18	214390846	Asian only (Age-adjusted)	2008	80.5%		77.7%
19	214390847	Native Hawaiian or Other Pacific Islander only (Age-adjusted)	2008	DSU		
20	214390848	2 or more races (Age-adjusted)	2008	75.6%		69.6%

**Figure 1.1** An Excel table is an example of structured data.



**Figure 1.2** Email is simultaneously an example of unstructured data and natural language data.

### 1.2.2 Unstructured data

Unstructured data is data that isn't easy to fit into a data model because the content is context-specific or varying. One example of unstructured data is your regular email (figure 1.2). Although email contains structured elements such as the sender, title, and body text, it's a challenge to find the number of people who have written an email complaint about a specific employee because so many ways exist to refer to a person, for example. The thousands of different languages and dialects out there further complicate this.

A human-written email, as shown in figure 1.2, is also a perfect example of natural language data.

### 1.2.3 Natural language

Natural language is a special type of unstructured data; it's challenging to process because it requires knowledge of specific data science techniques and linguistics.

The natural language processing community has had success in entity recognition, topic recognition, summarization, text completion, and sentiment analysis, but models trained in one domain don't generalize well to other domains. Even state-of-the-art techniques aren't able to decipher the meaning of every piece of text. This shouldn't be a surprise though: humans struggle with natural language as well. It's ambiguous by nature. The concept of meaning itself is questionable here. Have two people listen to the same conversation. Will they get the same meaning? The meaning of the same words can vary when coming from someone upset or joyous.

## 1.2.4 Machine-generated data

Machine-generated data is information that's automatically created by a computer, process, application, or other machine without human intervention. Machine-generated data is becoming a major data resource and will continue to do so. Wikibon has forecast that the market value of the *industrial Internet* (a term coined by Frost & Sullivan to refer to the integration of complex physical machinery with networked sensors and software) will be approximately \$540 billion in 2020. IDC (International Data Corporation) has estimated there will be 26 times more connected things than people in 2020. This network is commonly referred to as *the internet of things*.

The analysis of machine data relies on highly scalable tools, due to its high volume and speed. Examples of machine data are web server logs, call detail records, network event logs, and telemetry (figure 1.3).

CSIPERF:TXCOMMIT;313236		
2014-11-28 11:36:13, Info	CSI	00000153 Creating NT transaction (seq
69), objectname [6]"(null)"		
2014-11-28 11:36:13, Info	CSI	00000154 Created NT transaction (seq 69)
result 0x00000000, handle @0x4e54		
2014-11-28 11:36:13, Info	CSI	00000155@2014/11/28:10:36:13.471
Beginning NT transaction commit...		
2014-11-28 11:36:13, Info	CSI	00000156@2014/11/28:10:36:13.705 CSI perf
trace:		
CSIPERF:TXCOMMIT;273983		
2014-11-28 11:36:13, Info	CSI	00000157 Creating NT transaction (seq
70), objectname [6]"(null)"		
2014-11-28 11:36:13, Info	CSI	00000158 Created NT transaction (seq 70)
result 0x00000000, handle @0x4e5c		
2014-11-28 11:36:13, Info	CSI	00000159@2014/11/28:10:36:13.764
Beginning NT transaction commit...		
2014-11-28 11:36:14, Info	CSI	0000015a@2014/11/28:10:36:14.094 CSI perf
trace:		
CSIPERF:TXCOMMIT;386259		
2014-11-28 11:36:14, Info	CSI	0000015b Creating NT transaction (seq
71), objectname [6]"(null)"		
2014-11-28 11:36:14, Info	CSI	0000015c Created NT transaction (seq 71)
result 0x00000000, handle @0x4e5c		
2014-11-28 11:36:14, Info	CSI	0000015d@2014/11/28:10:36:14.106
Beginning NT transaction commit...		
2014-11-28 11:36:14, Info	CSI	0000015e@2014/11/28:10:36:14.428 CSI perf
trace:		
CSIPERF:TXCOMMIT;375581		

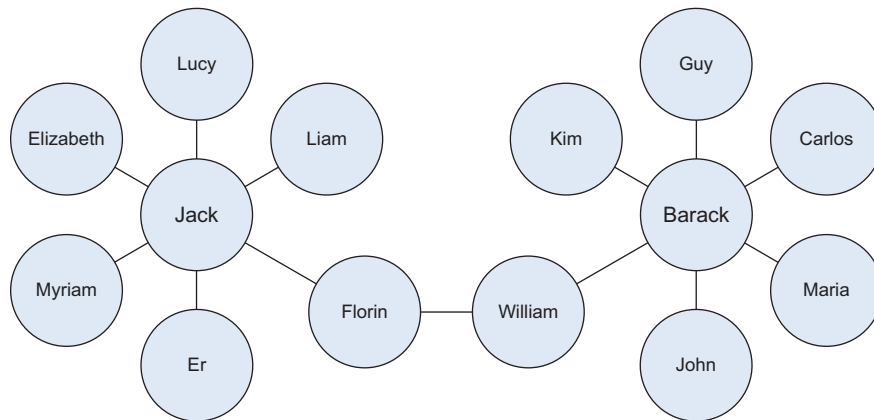
Figure 1.3 Example of machine-generated data

The machine data shown in figure 1.3 would fit nicely in a classic table-structured database. This isn't the best approach for highly interconnected or "networked" data, where the relationships between entities have a valuable role to play.

### 1.2.5 Graph-based or network data

"Graph data" can be a confusing term because any data can be shown in a graph. "Graph" in this case points to mathematical *graph theory*. In graph theory, a graph is a mathematical structure to model pair-wise relationships between objects. Graph or network data is, in short, data that focuses on the relationship or adjacency of objects. The graph structures use nodes, edges, and properties to represent and store graphical data. Graph-based data is a natural way to represent social networks, and its structure allows you to calculate specific metrics such as the influence of a person and the shortest path between two people.

Examples of graph-based data can be found on many social media websites (figure 1.4). For instance, on LinkedIn you can see who you know at which company. Your follower list on Twitter is another example of graph-based data. The power and sophistication comes from multiple, overlapping graphs of the same nodes. For example, imagine the connecting edges here to show "friends" on Facebook. Imagine another graph with the same people which connects business colleagues via LinkedIn. Imagine a third graph based on movie interests on Netflix. Overlapping the three different-looking graphs makes more interesting questions possible.



**Figure 1.4** Friends in a social network are an example of graph-based data.

Graph databases are used to store graph-based data and are queried with specialized query languages such as SPARQL.

Graph data poses its challenges, but for a computer interpreting additive and image data, it can be even more difficult.

### 1.2.6 **Audio, image, and video**

Audio, image, and video are data types that pose specific challenges to a data scientist. Tasks that are trivial for humans, such as recognizing objects in pictures, turn out to be challenging for computers. MLBAM (Major League Baseball Advanced Media) announced in 2014 that they'll increase video capture to approximately 7 TB per game for the purpose of live, in-game analytics. High-speed cameras at stadiums will capture ball and athlete movements to calculate in real time, for example, the path taken by a defender relative to two baselines.

Recently a company called DeepMind succeeded at creating an algorithm that's capable of learning how to play video games. This algorithm takes the video screen as input and learns to interpret everything via a complex process of deep learning. It's a remarkable feat that prompted Google to buy the company for their own Artificial Intelligence (AI) development plans. The learning algorithm takes in data as it's produced by the computer game; it's streaming data.

### 1.2.7 **Streaming data**

While streaming data can take almost any of the previous forms, it has an extra property. The data flows into the system when an event happens instead of being loaded into a data store in a batch. Although this isn't really a different type of data, we treat it here as such because you need to adapt your process to deal with this type of information.

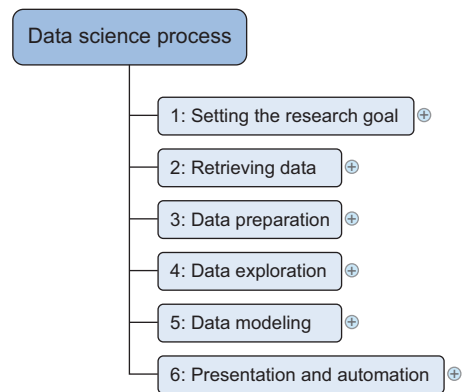
Examples are the "What's trending" on Twitter, live sporting or music events, and the stock market.

## 1.3 **The data science process**

The data science process typically consists of six steps, as you can see in the mind map in figure 1.5. We will introduce them briefly here and handle them in more detail in chapter 2.

### 1.3.1 **Setting the research goal**

Data science is mostly applied in the context of an organization. When the business asks you to perform a data science project, you'll first prepare a project charter. This charter contains information such as what you're going to research, how the company benefits from that, what data and resources you need, a timetable, and deliverables.



**Figure 1.5** The data science process



Throughout this book, the data science process will be applied to bigger case studies and you'll get an idea of different possible research goals.

### **1.3.2 Retrieving data**

The second step is to collect data. You've stated in the project charter which data you need and where you can find it. In this step you ensure that you can use the data in your program, which means checking the existence of, quality, and access to the data. Data can also be delivered by third-party companies and takes many forms ranging from Excel spreadsheets to different types of databases.

### **1.3.3 Data preparation**

Data collection is an error-prone process; in this phase you enhance the quality of the data and prepare it for use in subsequent steps. This phase consists of three sub-phases: *data cleansing* removes false values from a data source and inconsistencies across data sources, *data integration* enriches data sources by combining information from multiple data sources, and *data transformation* ensures that the data is in a suitable format for use in your models.

### **1.3.4 Data exploration**

Data exploration is concerned with building a deeper understanding of your data. You try to understand how variables interact with each other, the distribution of the data, and whether there are outliers. To achieve this you mainly use descriptive statistics, visual techniques, and simple modeling. This step often goes by the abbreviation EDA, for Exploratory Data Analysis.

### **1.3.5 Data modeling or model building**

In this phase you use models, domain knowledge, and insights about the data you found in the previous steps to answer the research question. You select a technique from the fields of statistics, machine learning, operations research, and so on. Building a model is an iterative process that involves selecting the variables for the model, executing the model, and model diagnostics.

### **1.3.6 Presentation and automation**

Finally, you present the results to your business. These results can take many forms, ranging from presentations to research reports. Sometimes you'll need to automate the execution of the process because the business will want to use the insights you gained in another project or enable an operational process to use the outcome from your model.

**AN ITERATIVE PROCESS** The previous description of the data science process gives you the impression that you walk through this process in a linear way, but in reality you often have to step back and rework certain findings. For instance, you might find outliers in the data exploration phase that point to data import errors. As part of the data science process you gain incremental insights, which may lead to new questions. To prevent rework, make sure that you scope the business question clearly and thoroughly at the start.

Now that we have a better understanding of the process, let's look at the technologies.

## **1.4 The big data ecosystem and data science**

Currently many big data tools and frameworks exist, and it's easy to get lost because new technologies appear rapidly. It's much easier once you realize that the big data ecosystem can be grouped into technologies that have similar goals and functionalities, which we'll discuss in this section. Data scientists use many different technologies, but not all of them; we'll dedicate a separate chapter to the most important data science technology classes. The mind map in figure 1.6 shows the components of the big data ecosystem and where the different technologies belong.

Let's look at the different groups of tools in this diagram and see what each does. We'll start with distributed file systems.

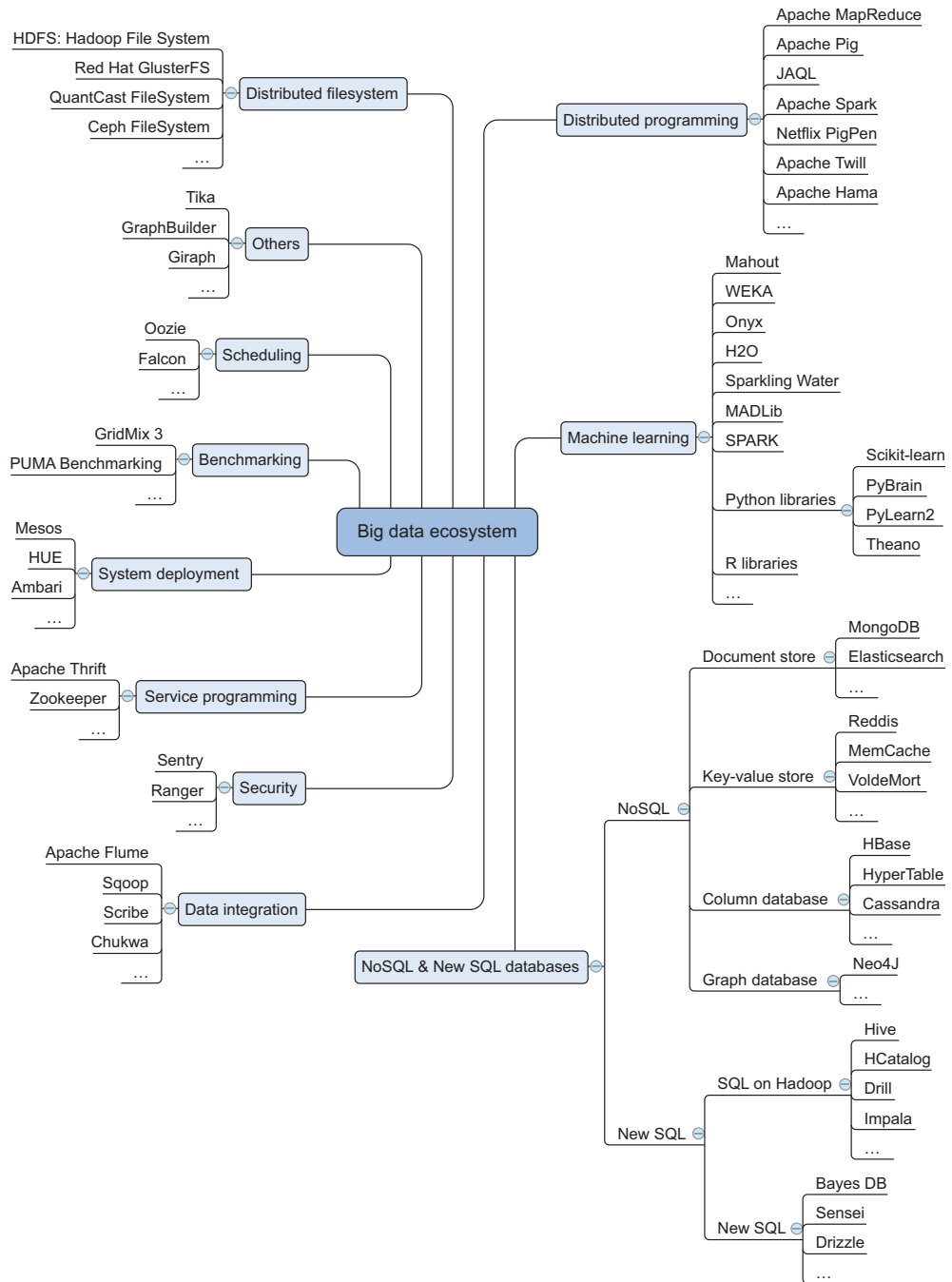
### **1.4.1 Distributed file systems**

A *distributed file system* is similar to a normal file system, except that it runs on multiple servers at once. Because it's a file system, you can do almost all the same things you'd do on a normal file system. Actions such as storing, reading, and deleting files and adding security to files are at the core of every file system, including the distributed one. Distributed file systems have significant advantages:

- They can store files larger than any one computer disk.
- Files get automatically replicated across multiple servers for redundancy or parallel operations while hiding the complexity of doing so from the user.
- The system scales easily: you're no longer bound by the memory or storage restrictions of a single server.

In the past, scale was increased by moving everything to a server with more memory, storage, and a better CPU (vertical scaling). Nowadays you can add another small server (horizontal scaling). This principle makes the scaling potential virtually limitless.

The best-known distributed file system at this moment is the *Hadoop File System (HDFS)*. It is an open source implementation of the Google File System. In this book we focus on the Hadoop File System because it is the most common one in use. However, many other distributed file systems exist: *Red Hat Cluster File System*, *Ceph File System*, and *Tachyon File System*, to name but three.



**Figure 1.6** Big data technologies can be classified into a few main components.

### **1.4.2 Distributed programming framework**

Once you have the data stored on the distributed file system, you want to exploit it. One important aspect of working on a distributed hard disk is that you won't move your data to your program, but rather you'll move your program to the data. When you start from scratch with a normal general-purpose programming language such as C, Python, or Java, you need to deal with the complexities that come with distributed programming, such as restarting jobs that have failed, tracking the results from the different subprocesses, and so on. Luckily, the open source community has developed many frameworks to handle this for you, and these give you a much better experience working with distributed data and dealing with many of the challenges it carries.

### **1.4.3 Data integration framework**

Once you have a distributed file system in place, you need to add data. You need to move data from one source to another, and this is where the data integration frameworks such as Apache Sqoop and Apache Flume excel. The process is similar to an extract, transform, and load process in a traditional data warehouse.

### **1.4.4 Machine learning frameworks**

When you have the data in place, it's time to extract the coveted insights. This is where you rely on the fields of machine learning, statistics, and applied mathematics. Before World War II everything needed to be calculated by hand, which severely limited the possibilities of data analysis. After World War II computers and scientific computing were developed. A single computer could do all the counting and calculations and a world of opportunities opened. Ever since this breakthrough, people only need to derive the mathematical formulas, write them in an algorithm, and load their data. With the enormous amount of data available nowadays, one computer can no longer handle the workload by itself. In fact, several algorithms developed in the previous millennium would never terminate before the end of the universe, even if you could use every computer available on Earth. This has to do with time complexity ([https://en.wikipedia.org/wiki/Time\\_complexity](https://en.wikipedia.org/wiki/Time_complexity)). An example is trying to break a password by testing every possible combination. An example can be found at <http://stackoverflow.com/questions/7055652/real-world-example-of-exponential-time-complexity>. One of the biggest issues with the old algorithms is that they don't scale well. With the amount of data we need to analyze today, this becomes problematic, and specialized frameworks and libraries are required to deal with this amount of data. The most popular machine-learning library for Python is Scikit-learn. It's a great machine-learning toolbox, and we'll use it later in the book. There are, of course, other Python libraries:

- *PyBrain for neural networks*—Neural networks are learning algorithms that mimic the human brain in learning mechanics and complexity. Neural networks are often regarded as advanced and black box.

- *NLTK or Natural Language Toolkit*—As the name suggests, its focus is working with natural language. It's an extensive library that comes bundled with a number of text corpuses to help you model your own data.
- *Pylearn2*—Another machine learning toolbox but a bit less mature than Scikit-learn.
- *TensorFlow*—A Python library for deep learning provided by Google.

The landscape doesn't end with Python libraries, of course. Spark is a new Apache-licensed machine-learning engine, specializing in real-time machine learning. It's worth taking a look at and you can read more about it at <http://spark.apache.org/>.

### 1.4.5 NoSQL databases

If you need to store huge amounts of data, you require software that's specialized in managing and querying this data. Traditionally this has been the playing field of relational databases such as Oracle SQL, MySQL, Sybase IQ, and others. While they're still the go-to technology for many use cases, new types of databases have emerged under the grouping of NoSQL databases.

The name of this group can be misleading, as “No” in this context stands for “Not Only.” A lack of functionality in SQL isn't the biggest reason for the paradigm shift, and many of the NoSQL databases have implemented a version of SQL themselves. But traditional databases had shortcomings that didn't allow them to scale well. By solving several of the problems of traditional databases, NoSQL databases allow for a virtually endless growth of data. These shortcomings relate to every property of big data: their storage or processing power can't scale beyond a single node and they have no way to handle streaming, graph, or unstructured forms of data.

Many different types of databases have arisen, but they can be categorized into the following types:

- *Column databases*—Data is stored in columns, which allows algorithms to perform much faster queries. Newer technologies use cell-wise storage. Table-like structures are still important.
- *Document stores*—Document stores no longer use tables, but store every observation in a document. This allows for a much more flexible data scheme.
- *Streaming data*—Data is collected, transformed, and aggregated not in batches but in real time. Although we've categorized it here as a database to help you in tool selection, it's more a particular type of problem that drove creation of technologies such as Storm.
- *Key-value stores*—Data isn't stored in a table; rather you assign a key for every value, such as `org.marketing.sales.2015: 20000`. This scales well but places almost all the implementation on the developer.
- *SQL on Hadoop*—Batch queries on Hadoop are in a SQL-like language that uses the map-reduce framework in the background.
- *New SQL*—This class combines the scalability of NoSQL databases with the advantages of relational databases. They all have a SQL interface and a relational data model.

- *Graph databases*—Not every problem is best stored in a table. Particular problems are more naturally translated into graph theory and stored in graph databases. A classic example of this is a social network.

#### **1.4.6 Scheduling tools**

Scheduling tools help you automate repetitive tasks and trigger jobs based on events such as adding a new file to a folder. These are similar to tools such as CRON on Linux but are specifically developed for big data. You can use them, for instance, to start a MapReduce task whenever a new dataset is available in a directory.

#### **1.4.7 Benchmarking tools**

This class of tools was developed to optimize your big data installation by providing standardized profiling suites. A profiling suite is taken from a representative set of big data jobs. Benchmarking and optimizing the big data infrastructure and configuration aren't often jobs for data scientists themselves but for a professional specialized in setting up IT infrastructure; thus they aren't covered in this book. Using an optimized infrastructure can make a big cost difference. For example, if you can gain 10% on a cluster of 100 servers, you save the cost of 10 servers.

#### **1.4.8 System deployment**

Setting up a big data infrastructure isn't an easy task and assisting engineers in deploying new applications into the big data cluster is where system deployment tools shine. They largely automate the installation and configuration of big data components. This isn't a core task of a data scientist.

#### **1.4.9 Service programming**

Suppose that you've made a world-class soccer prediction application on Hadoop, and you want to allow others to use the predictions made by your application. However, you have no idea of the architecture or technology of everyone keen on using your predictions. Service tools excel here by exposing big data applications to other applications as a service. Data scientists sometimes need to expose their models through services. The best-known example is the REST service; REST stands for representational state transfer. It's often used to feed websites with data.

#### **1.4.10 Security**

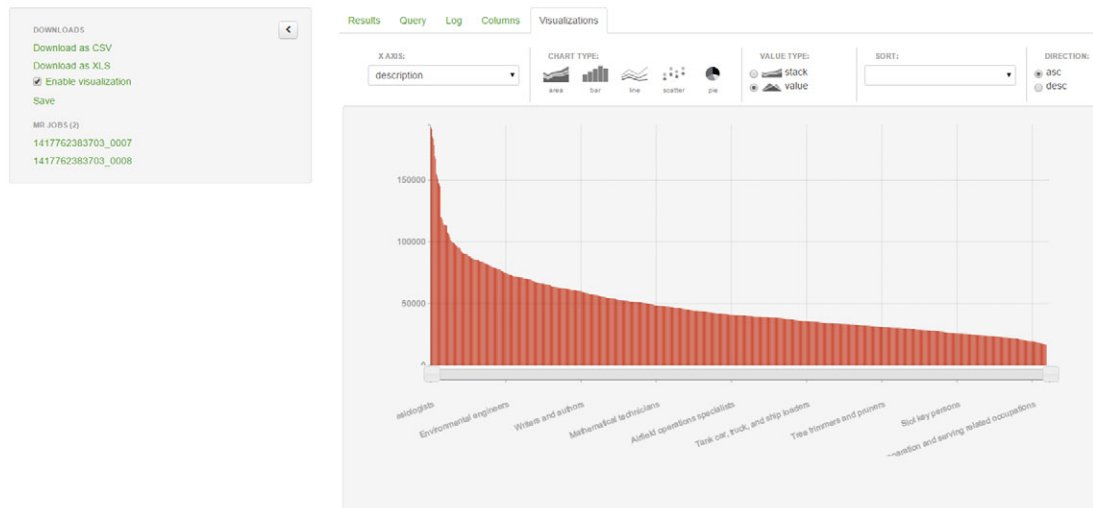
Do you want everybody to have access to all of your data? You probably need to have fine-grained control over the access to data but don't want to manage this on an application-by-application basis. Big data security tools allow you to have central and fine-grained control over access to the data. Big data security has become a topic in its own right, and data scientists are usually only confronted with it as data consumers; seldom will they implement the security themselves. In this book we don't describe how to set up security on big data because this is a job for the security expert.

## 1.5 An introductory working example of Hadoop

We'll end this chapter with a small application in a big data context. For this we'll use a Hortonworks Sandbox image. This is a virtual machine created by Hortonworks to try some big data applications on a local machine. Later on in this book you'll see how Juju eases the installation of Hadoop on multiple machines.

We'll use a small data set of job salary data to run our first sample, but querying a large data set of billions of rows would be equally easy. The query language will seem like SQL, but behind the scenes a MapReduce job will run and produce a straightforward table of results, which can then be turned into a bar graph. The end result of this exercise looks like figure 1.7.

### Query Results: Unsaved Query



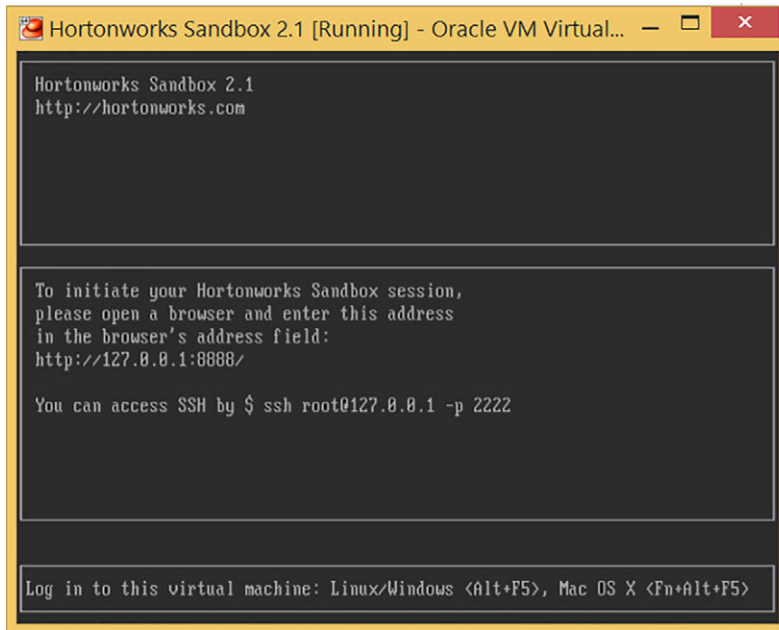
**Figure 1.7** The end result: the average salary by job description

To get up and running as fast as possible we use a Hortonworks Sandbox inside VirtualBox. VirtualBox is a virtualization tool that allows you to run another operating system inside your own operating system. In this case you can run CentOS with an existing Hadoop installation inside your installed operating system.

A few steps are required to get the sandbox up and running on VirtualBox. Caution, the following steps were applicable at the time this chapter was written (February 2015):

- 1 Download the virtual image from <http://hortonworks.com/products/hortonworks-sandbox/#install>.
- 2 Start your virtual machine host. VirtualBox can be downloaded from <https://www.virtualbox.org/wiki/Downloads>.

- 3 Press CTRL+I and select the virtual image from Hortonworks.
- 4 Click Next.
- 5 Click Import; after a little time your image should be imported.
- 6 Now select your virtual machine and click Run.
- 7 Give it a little time to start the CentOS distribution with the Hadoop installation running, as shown in figure 1.8. Notice the Sandbox version here is 2.1. With other versions things could be slightly different.

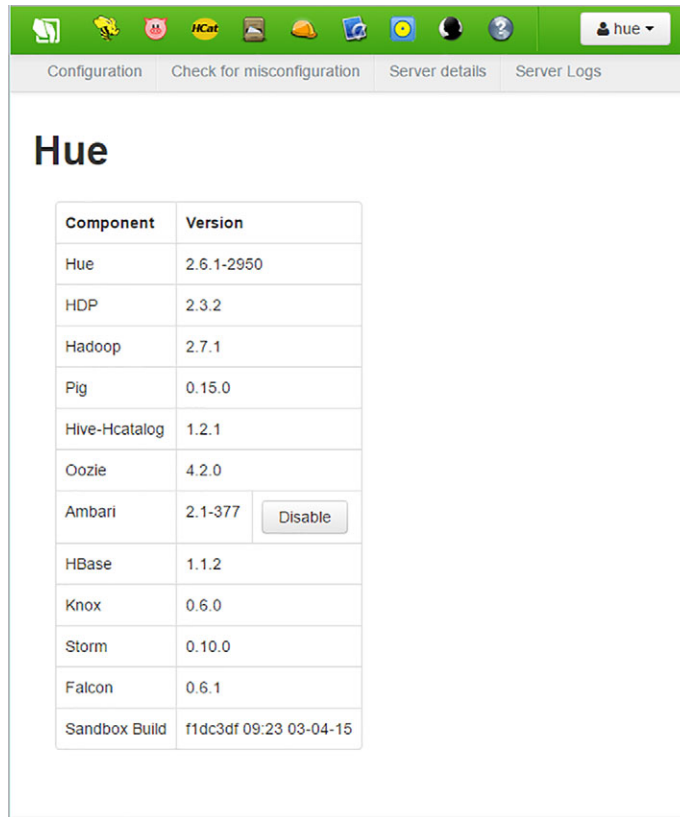


**Figure 1.8** Hortonworks Sandbox running within VirtualBox

You can directly log on to the machine or use SSH to log on. For this application you'll use the web interface. Point your browser to the address <http://127.0.0.1:8000> and you'll be welcomed with the screen shown in figure 1.9.

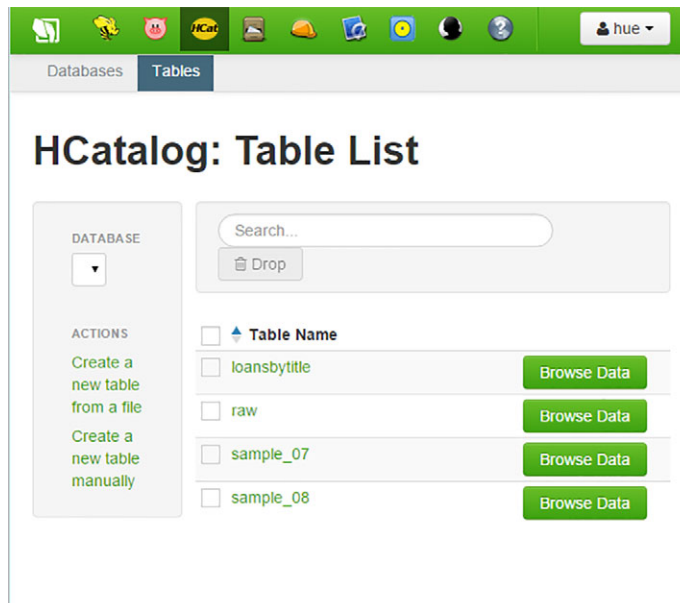
Hortonworks has uploaded two sample sets, which you can see in HCatalog. Just click the HCat button on the screen and you'll see the tables available to you (figure 1.10).





Component	Version	
Hue	2.6.1-2950	
HDP	2.3.2	
Hadoop	2.7.1	
Pig	0.15.0	
Hive-Hcatalog	1.2.1	
Oozie	4.2.0	
Ambari	2.1-377	<button>Disable</button>
HBase	1.1.2	
Knox	0.6.0	
Storm	0.10.0	
Falcon	0.6.1	
Sandbox Build	f1dc3df 09:23 03-04-15	

**Figure 1.9** The Hortonworks Sandbox welcome screen available at <http://127.0.0.1:8000>



**HCatalog: Table List**

DATABASE: ▼

ACTIONS:

- Create a new table from a file
- Create a new table manually

Search...

Drop

<input type="checkbox"/> Table Name	
<input type="checkbox"/> loansbytitle	<button>Browse Data</button>
<input type="checkbox"/> raw	<button>Browse Data</button>
<input type="checkbox"/> sample_07	<button>Browse Data</button>
<input type="checkbox"/> sample_08	<button>Browse Data</button>

**Figure 1.10** A list of available tables in HCatalog

Query Results: **sample\_07**

DOWNLOADS

Download as CSV

Download as XLSX

Save

Did you know? If the result contains a large number of columns, click a row to select a column to jump to. As you type into the field, a drop-down list displays column names that match the string.

	default.sample_07.code	default.sample_07.description	default.sample_07
0	00-0000	All Occupations	134354250
1	11-0000	Management occupations	6003930
2	11-1011	Chief executives	299160
3	11-1021	General and operations managers	1655410
4	11-1031	Legislators	61110
5	11-2011	Advertising and promotions managers	36300
6	11-2021	Marketing managers	165240
7	11-2022	Sales managers	322170
8	11-2031	Public relations managers	47210
9	11-3011	Administrative services managers	239360
10	11-3021	Computer and information systems managers	264990
11	11-3031	Financial managers	484390
12	11-3041	Compensation and benefits managers	41780
13	11-3042	Training and development managers	28170
14	11-3049	Human resources managers, all other	58100
15	11-3051	Industrial production managers	152870
16	11-3061	Purchasing managers	65600
17	11-3071	Transportation, storage, and distribution managers	92790
18	11-9011	Farm, ranch, and other agricultural managers	3480

Next Page →

**Figure 1.11** The contents of the table

To see the contents of the data, click the Browse Data button next to the sample\_07 entry to get the next screen (figure 1.11).

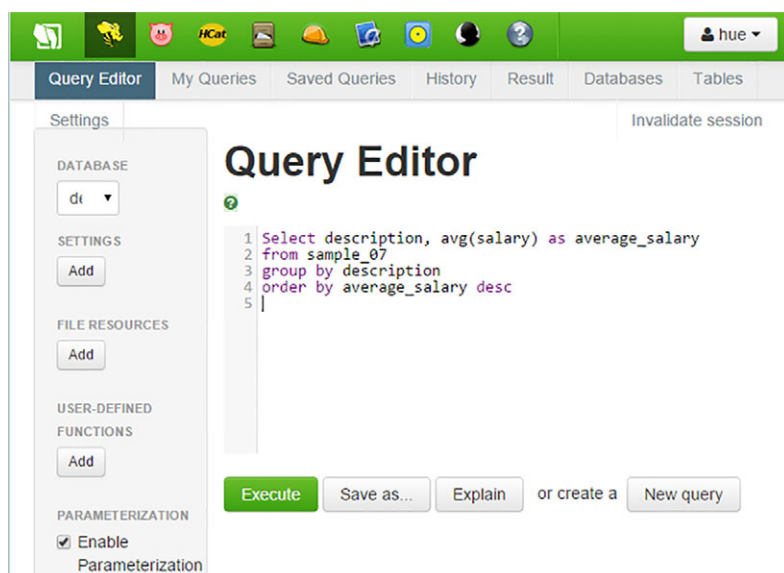
This looks like an ordinary table, and Hive is a tool that lets you approach it like an ordinary database with SQL. That's right: in Hive you get your results using HiveQL, a dialect of plain-old SQL. To open the Beeswax HiveQL editor, click the Beeswax button in the menu (figure 1.12).

To get your results, execute the following query:

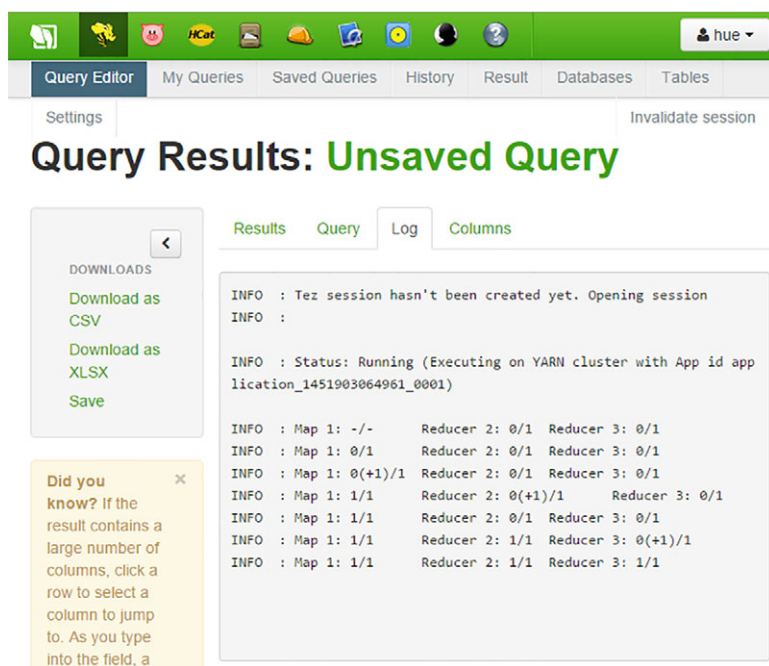
```
Select description, avg(salary) as average_salary from sample_07 group by
description order by average_salary desc.
```

Click the Execute button. Hive translates your HiveQL into a MapReduce job and executes it in your Hadoop environment, as you can see in figure 1.13.

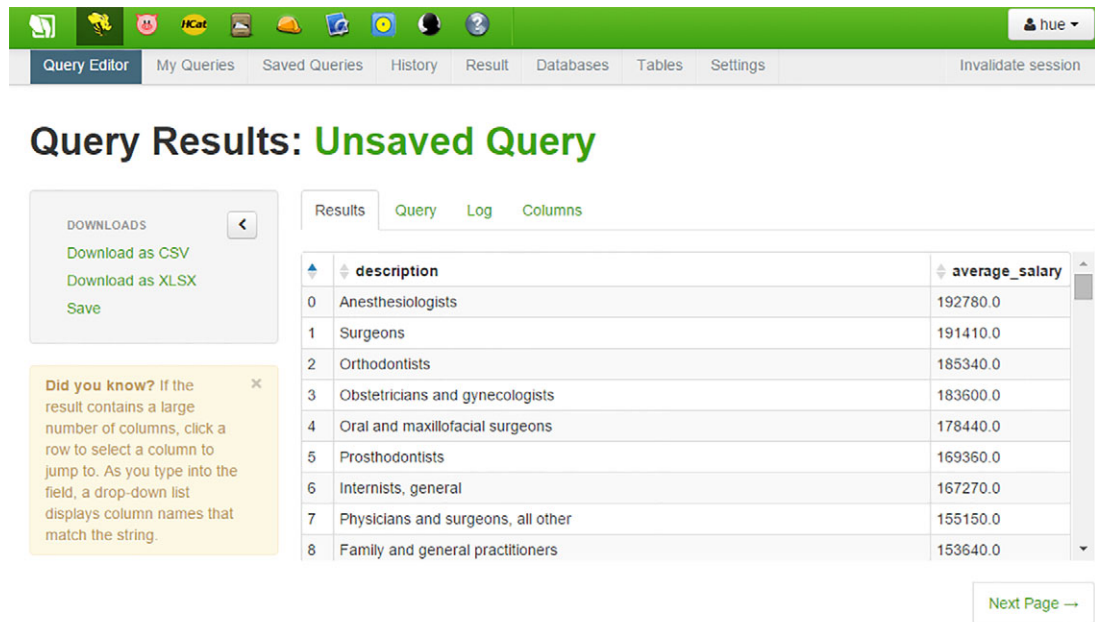
Best however to avoid reading the log window for now. At this point, it's misleading. If this is your first query, then it could take 30 seconds. Hadoop is famous for its warming periods. That discussion is for later, though.



**Figure 1.12** You can execute a HiveQL command in the Beeswax HiveQL editor. Behind the scenes it's translated into a MapReduce job.



**Figure 1.13** The logging shows that your HiveQL is translated into a MapReduce job. Note: This log was from the February 2015 version of HDP, so the current version might look slightly different.



**Figure 1.14** The end result: an overview of the average salary by profession

After a while the result appears. Great work! The conclusion of this, as shown in figure 1.14, is that going to medical school is a good investment. Surprised?

With this table we conclude our introductory Hadoop tutorial.

Although this chapter was but the beginning, it might have felt a bit overwhelming at times. It's recommended to leave it be for now and come back here again when all the concepts have been thoroughly explained. Data science is a broad field so it comes with a broad vocabulary. We hope to give you a glimpse of most of it during our time together. Afterward, you pick and choose and hone your skills in whatever direction interests you the most. That's what "Introducing Data Science" is all about and we hope you'll enjoy the ride with us.

## 1.6 Summary

In this chapter you learned the following:

- *Big data* is a blanket term for any collection of data sets so large or complex that it becomes difficult to process them using traditional data management techniques. They are characterized by the four Vs: velocity, variety, volume, and veracity.
- *Data science* involves using methods to analyze small data sets to the gargantuan ones big data is all about.

- Even though the *data science process* isn't linear it can be divided into steps:
  - 1 Setting the research goal
  - 2 Gathering data
  - 3 Data preparation
  - 4 Data exploration
  - 5 Modeling
  - 6 Presentation and automation
- The big data landscape is more than Hadoop alone. It consists of many different technologies that can be categorized into the following:
  - File system
  - Distributed programming frameworks
  - Data integration
  - Databases
  - Machine learning
  - Security
  - Scheduling
  - Benchmarking
  - System deployment
  - Service programming
- Not every big data category is utilized heavily by data scientists. They focus mainly on the file system, the distributed programming frameworks, databases, and machine learning. They do come in contact with the other components, but these are domains of other professions.
- Data can come in different forms. The main forms are
  - Structured data
  - Unstructured data
  - Natural language data
  - Machine data
  - Graph-based data
  - Streaming data