5 Primeiros passos em big data

Este capítulo cobre

- ÿ Dando os primeiros passos com dois aplicativos de big data: Hadoop e Spark ÿ Usando
- Python para escrever trabalhos de big data ÿ
- Construindo um painel interativo que se conecta aos dados armazenados em um banco de dados de big data

Nos últimos dois capítulos, aumentamos constantemente o tamanho dos dados. No capítulo 3 trabalhamos com conjuntos de dados que poderiam caber na memória principal de um computador. O Capítulo 4 introduziu técnicas para lidar com conjuntos de dados que eram grandes demais para caber na memória, mas que ainda podiam ser processados em um único computador. Neste capítulo você aprenderá a trabalhar com tecnologias que podem lidar com dados tão grandes que um único nó (computador) não é mais suficiente. Na verdade, pode nem caber em cem computadores. Agora isso é um desafio, não é?

Ficaremos o mais próximo possível da forma de trabalhar dos capítulos anteriores; o foco é dar a você a confiança necessária para trabalhar em uma plataforma de big data. Para fazer isso, a parte principal deste capítulo é um estudo de caso. Você criará um painel que permite

você explorar dados de credores de um banco. Ao final deste capítulo você terá passou pelas seguintes etapas:

- ÿ Carregue dados no Hadoop, a plataforma de big data mais comum.
- ÿ Transforme e limpe dados com o Spark.
- ÿ Armazene-os em um banco de dados de big data chamado Hive.
- ÿ Visualize esses dados de forma interativa com o Qlik Sense, uma ferramenta de visualização.

Tudo isso (além da visualização) será coordenado dentro de um Python roteiro. O resultado final é um painel que permite explorar os dados, conforme mostrado em figura 5.1.

Figura 5.1 Painel interativo do Qlik

Tenha em mente que iremos apenas arranhar a superfície da prática e da teoria neste capítulo introdutório sobre tecnologias de big data. O estudo de caso abordará três grandes tecnologias de dados (Hadoop, Spark e Hive), mas apenas para manipulação de dados, não construção de modelo. Caberá a você combinar as tecnologias de big data que você conseguir veja aqui as técnicas de construção de modelos que abordamos nos capítulos anteriores.

5.1 Distribuindo armazenamento e processamento de dados com estruturas

Novas tecnologias de big data, como Hadoop e Spark, facilitam muito o trabalho com e controlar um cluster de computadores. O Hadoop pode escalar até milhares de computadores, criando um cluster com petabytes de armazenamento. Isso permite que as empresas compreendam o valor da enorme quantidade de dados disponíveis.

5.1.1 Hadoop: uma estrutura para armazenar e processar grandes conjuntos de dados

Apache Hadoop é uma estrutura que simplifica o trabalho com um cluster de computadores. O objetivo é ser todas as seguintes coisas e muito mais:

- ÿ *Confiável* — Criando automaticamente diversas cópias dos dados e reimplantando a lógica de processamento em caso de falha.
- ÿ *Tolerante a falhas — detecta* falhas e aplica recuperação automática. ÿ *Escalável*
- — *os dados* e seu processamento são distribuídos em clusters de computadores (escala horizontal). ÿ
- Portátil — Instalável em todos os tipos de hardware e sistemas operacionais.

A estrutura principal é composta por um sistema de arquivos distribuído, um gerenciador de recursos e um sistema para executar programas distribuídos. Na prática, ele permite que você trabalhe com o sistema de arquivos distribuído quase tão facilmente quanto com o sistema de arquivos local do seu computador doméstico. Mas, em segundo plano, os dados podem estar espalhados entre milhares de servidores.

OS DIFERENTES COMPONENTES DO HADOOP

No coração do Hadoop encontramos

- ÿ Um sistema de arquivos distribuído
- (HDFS) ÿ Um método para executar programas em grande escala (MapReduce) ÿ
- Um sistema para gerenciar os recursos do cluster (YARN)

Além disso, surgiu um ecossistema de aplicações (figura 5.2), como os bancos de dados Hive e HBase e estruturas para aprendizado de máquina, como o Mahout. Usaremos o Hive neste capítulo. O Hive possui uma linguagem baseada no SQL amplamente utilizado para interagir com os dados armazenados no banco de dados.

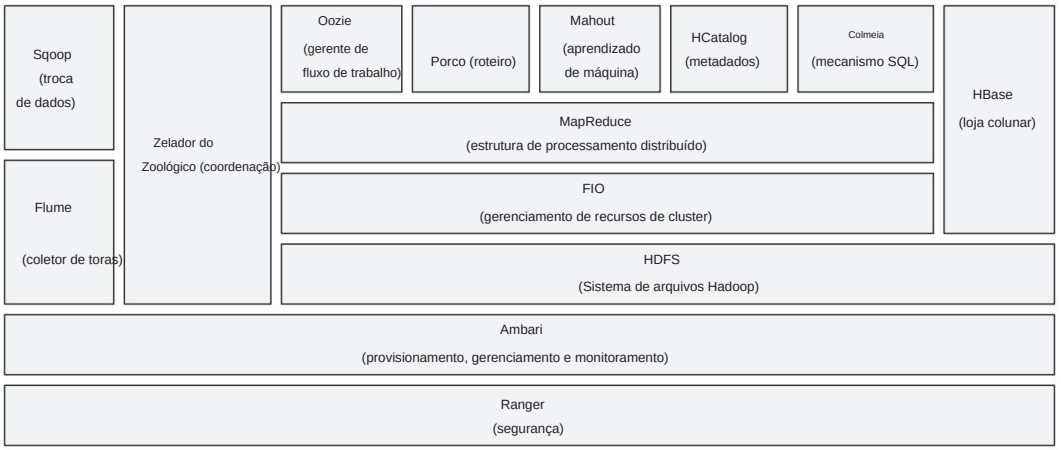


Figura 5.2 Uma amostra do ecossistema de aplicativos que surgiu em torno do Hadoop Core Framework

É possível usar a popular ferramenta Impala para consultar dados do Hive até 100 vezes mais rápido. Não entraremos no Impala neste livro, mas mais informações podem ser encontradas em http://impala.io/. Já tivemos uma breve introdução ao MapReduce no capítulo 4, mas vamos elaborar um pouco aqui porque é uma parte vital do Hadoop.

MAPREDUCE: COMO O HADOOP ALCANÇA O PARALELISMO

O Hadoop usa um método de programação chamado MapReduce para obter paralelismo. A O algoritmo MapReduce divide os dados, processa-os em paralelo e, em seguida, classifica, combina e agrega os resultados novamente. No entanto, o algoritmo MapReduce não é adequado para análises interativas ou programas iterativos porque grava os dados para um disco entre cada etapa computacional. Isso é caro quando se trabalha com grandes conjuntos de dados.

Vamos ver como o MapReduce funcionaria em um pequeno exemplo fictício. Você é o diretor de uma empresa de brinquedos. Cada brinquedo tem duas cores, e quando um cliente encomenda um brinquedo da página da web, a página da web coloca um arquivo de pedido no Hadoop com as cores do brinquedo. Sua tarefa é descobrir quantas unidades de cores você precisa preparar. Você usará um Algoritmo estilo MapReduce para contar as cores. Primeiro vamos dar uma olhada em uma versão simplificada na figura 5.3.

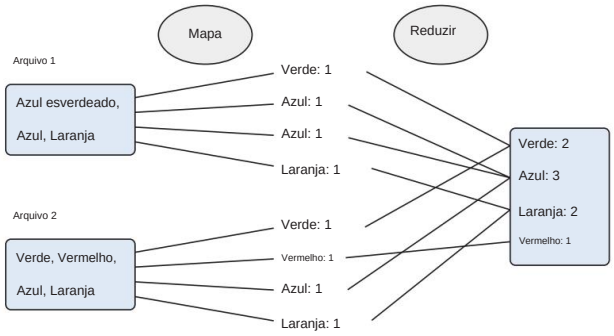


Figura 5.3 Um exemplo simplificado de fluxo MapReduce para contagem de cores em textos de entrada

Como o nome sugere, o processo se resume basicamente em duas grandes fases:

- *Fase de mapeamento — Os* documentos são divididos em pares de valores-chave. Até nós reduzir, podemos ter muitas duplicatas.
- Fase *de redução* — Não é diferente de um “agrupar por” SQL . As diferentes ocorrências únicas são agrupadas e dependendo da função redutora, um resultado diferente pode ser criado. Aqui queríamos uma contagem por cor, então é isso que reduzir retornos de função.

Na realidade, é um pouco mais complicado do que isso.

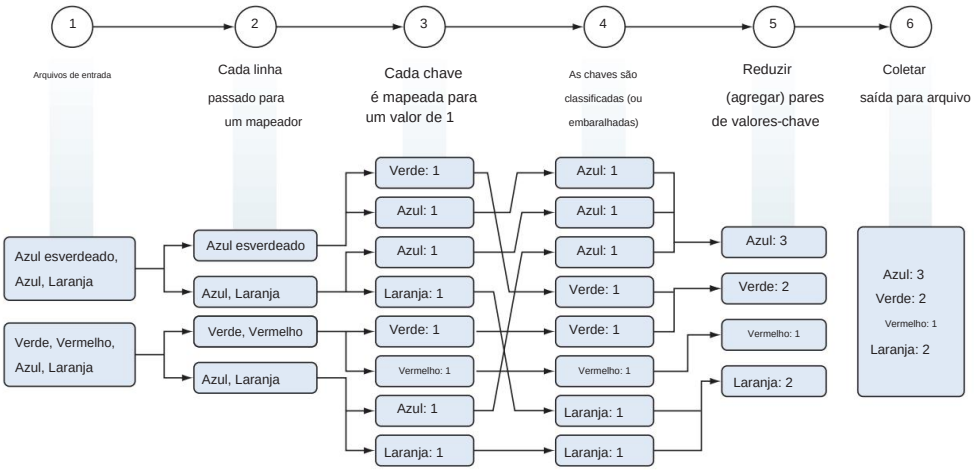


Figura 5.4 Um exemplo de fluxo MapReduce para contagem de cores em textos de entrada

Todo o processo é descrito nas seis etapas a seguir e representado na figura 5.4.

1 Lendo os arquivos de entrada.

2 Passando cada linha para um trabalho de mapeador.

3 A tarefa do mapeador analisa as cores (chaves) do arquivo e gera um arquivo para cada cor com o número de vezes que ela foi encontrada (valor). Ou, mais tecnicamente, mapeia uma chave (a cor) para um valor (o número de ocorrências).

4 As chaves são embaralhadas e classificadas para facilitar a agregação.

5 A fase de redução soma o número de ocorrências por cor e gera uma arquivo por chave com o número total de ocorrências de cada cor.

6 As chaves são coletadas em um arquivo de saída.

NOTA Embora o Hadoop facilite o trabalho com big data, configurar um bom cluster funcional ainda não é trivial, mas gerenciadores de cluster como o Apache Mesos aliviam a carga. Na realidade, muitas empresas (de médio porte) não têm competência para manter uma instalação Hadoop saudável. É por isso que trabalharemos com o Hortonworks Sandbox, um ecossistema Hadoop pré-instalado e configurado. As instruções de instalação podem ser encontradas na seção 1.5: Um exemplo introdutório de trabalho do Hadoop.

Agora, tendo em mente o funcionamento do Hadoop, vamos dar uma olhada no Spark.

5.1.2 Spark: substituindo MapReduce para melhor desempenho

Os cientistas de dados geralmente fazem análises interativas e contam com algoritmos que são inerentemente iterativos; pode demorar um pouco até que um algoritmo convirja para uma solução. Como este é um ponto fraco do framework MapReduce, apresentaremos o Spark Framework para superá-lo. O Spark melhora o desempenho nessas tarefas em uma ordem de grandeza.

O QUE É FAÍSCA?

Spark é uma estrutura de computação em cluster semelhante ao MapReduce. O Spark, no entanto, não cuida do armazenamento de arquivos no próprio sistema de arquivos (distribuído), nem do gerenciamento de recursos. Para isso conta com sistemas como Hadoop File System, YARN ou Apache Mesos. Hadoop e Spark são, portanto, sistemas complementares. Para teste e desenvolvimento, você pode até executar o Spark em seu sistema local.

COMO O SPARK RESOLVE OS PROBLEMAS DO MAPREDUCE?

Embora simplifiquemos um pouco as coisas para maior clareza, o Spark cria uma espécie de memória RAM compartilhada entre os computadores do seu cluster. Isto permite que diferentes trabalhadores compartilhem variáveis (e seu estado) e, assim, elimina a necessidade de gravar os resultados intermediários no disco. Mais tecnicamente e mais corretamente se você gosta disso: o Spark usa Resilient Distributed Datasets (RDD), que são uma abstração de memória distribuída que permite aos programadores realizar cálculos na memória em grandes clusters de maneira tolerante a falhas . -sistema de memória, evita operações dispendiosas de disco.

OS DIFERENTES COMPONENTES DO ECOSSISTEMA SPARK

O núcleo Spark fornece um ambiente NoSQL adequado para análises exploratórias e interativas. O Spark pode ser executado em lote e em modo interativo e oferece suporte a Python.

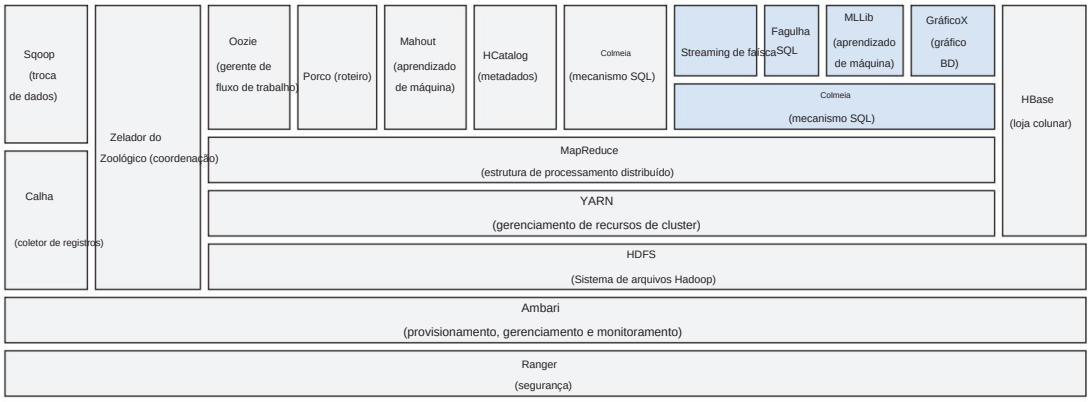


Figura 5.5 A estrutura Spark quando usada em combinação com a estrutura Hadoop

O Spark possui quatro outros componentes grandes, listados abaixo e representados na figura 5.5.

1 Spark streaming é uma ferramenta para análise em tempo real.

2 Spark SQL fornece uma interface SQL para trabalhar com Spark.

3 MLLib é uma ferramenta para aprendizado de máquina dentro da estrutura Spark.

4 GraphX é um banco de dados gráfico para Spark. Iremos nos aprofundar nos bancos de dados gráficos em Capítulo 7.

---

1 Consulte https://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf.

Agora vamos mergulhar nos dados de empréstimos usando Hadoop, Hive e Spark.

5.2 Estudo de caso: Avaliando o risco ao emprestar dinheiro

Enriquecidos com um conhecimento básico de Hadoop e Spark, agora estamos prontos para colocar a mão na massa em big data. O objetivo deste estudo de caso é ter uma primeira experiência com as tecnologias que apresentamos anteriormente neste capítulo e ver que, em grande parte, você pode (mas não precisa) trabalhar de forma semelhante a outras tecnologias. Observação: a parte dos dados usada aqui não é tão grande porque isso exigiria muita largura de banda para coletá-los e vários nós para acompanhar o exemplo.

O que usaremos

ÿ Horton Sandbox em uma máquina virtual. Se você não baixou e importou isso para um software VM como o VirtualBox, volte para a seção 1.5, onde isso é explicado. A versão 2.3.2 do Horton Sandbox foi usada ao escrever este capítulo.

ÿ Bibliotecas Python: Pandas e pywebhdsf. Desta vez, eles não precisam ser instalados em seu ambiente virtual local; precisamos deles diretamente no Horton Sandbox. Portanto, precisamos iniciar o Horton Sandbox (no Virtu-alBox, por exemplo) e fazer alguns preparativos.

Na linha de comando do Sandbox ainda há várias coisas que você precisa fazer para que tudo isso funcione, então conecte-se à linha de comando. Você pode fazer isso usando um programa como o PuTTY.

Se você não estiver familiarizado com o PuTTY, ele oferece uma interface de linha de comando para servidores e pode ser baixado gratuitamente em http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html .

A configuração de login do PuTTY é mostrada na figura 5.6.

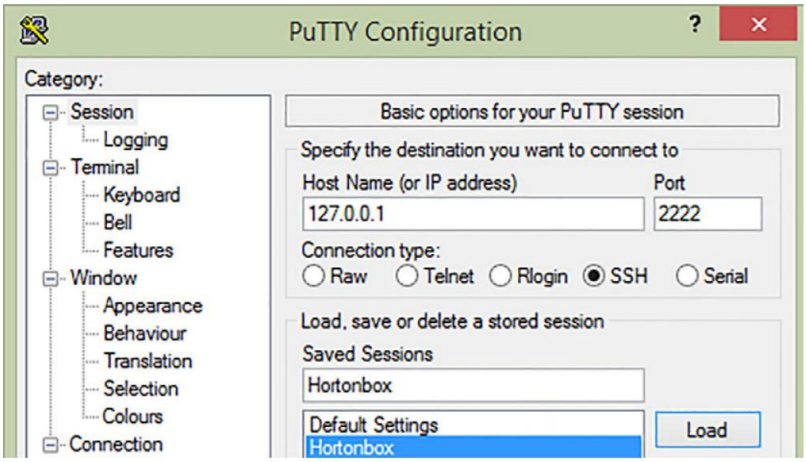


Figura 5.6 Conectando ao Horton Sandbox usando PuTTY

O usuário e a senha padrão são (no momento da escrita) “root” e “hadoop”, respectivamente. No entanto, você precisará alterar essa senha no primeiro login.

Uma vez conectado, emita os seguintes comandos: ÿ yum

-y install python-pip — Isso instala o pip, um gerenciador de pacotes Python. ÿ pip install git+https://github.com/DavyCielen/pywebhdfs.git –upgrade— No momento em que este artigo foi escrito, houve um problema com a biblioteca pywebhdfs e corrigimos isso nesta bifurcação. Esperamos que você não precise mais disso quando ler isto; o problema foi sinalizado e deverá ser resolvido pelos mantenedores deste pacote.

ÿ pip install pandas — Para instalar o Pandas. Isso geralmente demora um pouco por causa do dependências.

Um arquivo .ipynb está disponível para você abrir no Jupyter ou no Ipython (o mais antigo) e acompanhar o código neste capítulo. As instruções de configuração do Horton Sandbox são repetidas aqui; certifique-se de executar o código diretamente no Horton Sandbox. Agora, com os assuntos preparatórios resolvidos, vamos ver o que precisaremos fazer.

Neste exercício, passaremos por várias outras etapas do processo de ciência de dados:

Etapa 1: O objetivo da pesquisa. Isso consiste em duas partes:

ÿ Fornecer um painel ao nosso gerente ÿ Preparar dados para que outras pessoas criem seus próprios painéis

Etapa 2: recuperação de dados

ÿ Baixar os dados do site do clube de empréstimo ÿ Colocar os dados no sistema de arquivos Hadoop do Horton Sandbox

Etapa 3: Preparação de dados

ÿ Transformando esses dados com Spark ÿ Armazenando os dados preparados no Hive

Etapas 4 e 6: Exploração e criação de relatórios ÿ

Visualizando os dados com o Qlik Sense

Não temos construção de modelo neste estudo de caso, mas você terá a infraestrutura para fazer isso sozinho, se desejar. Por exemplo, você pode usar o aprendizado de máquina SPARK para tentar prever quando alguém deixará de pagar sua dívida.

É hora de conhecer o Lending Club.

5.2.1 Passo 1: O objetivo da pesquisa O

Lending Club é uma organização que conecta pessoas que precisam de um empréstimo com pessoas que têm dinheiro para investir. Seu chefe também tem dinheiro para investir e quer informações antes de jogar uma quantia substancial na mesa. Para conseguir isso, você criará um relatório para ele que lhe dará informações sobre a classificação média, os riscos e o retorno do empréstimo de dinheiro a uma determinada pessoa. Ao passar por esse processo, você torna os dados

acessível em uma ferramenta de painel, permitindo assim que outras pessoas também o explorem. Em um sentido que este é o objetivo secundário deste caso: abrir os dados para BI de autoatendimento .

O Business Intelligence de autoatendimento é frequentemente aplicado em organizações orientadas por dados que não temos analistas de sobra. Qualquer pessoa na organização pode fazer o simples fatiamento e se analisando e deixando as análises mais complicadas para o cientista de dados.

Podemos fazer este estudo de caso porque o Lending Club disponibiliza dados anônimos sobre os empréstimos existentes. Ao final deste estudo de caso, você criará um relatório semelhante à figura 5.7.

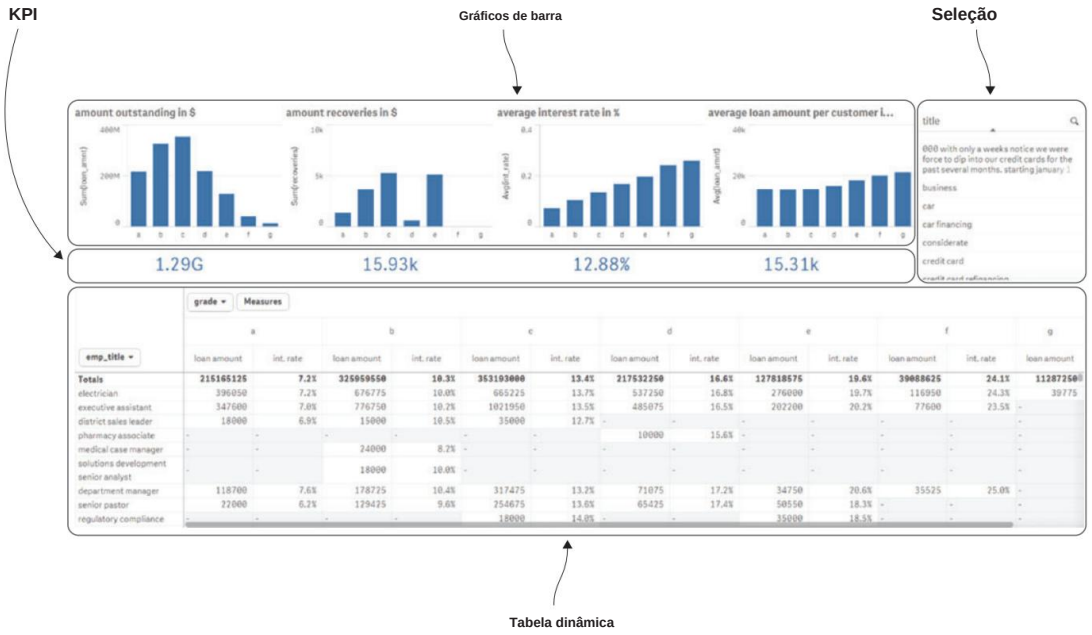


Figura 5.7 O resultado final deste exercício é um painel explicativo para comparar uma oportunidade de empréstimo com oportunidades semelhantes.

Comecemos pelo princípio; vamos obter dados.

5.2.2 Passo 2: Recuperação de dados

É hora de trabalhar com o sistema de arquivos Hadoop (ou hdfs). Primeiro enviaremos comandos através da linha de comando e, em seguida, através da linguagem de script Python com o ajuda do pacote pywebhdfs.

O sistema de arquivos Hadoop é semelhante a um sistema de arquivos normal, exceto que os arquivos e pastas são armazenadas em vários servidores e você não sabe o endereço físico de cada arquivo. Isso não é estranho se você já trabalhou com ferramentas como Dropbox ou Google Dirigir. Os arquivos que você coloca nessas unidades são armazenados em algum lugar de um servidor sem você

sabendo exatamente em qual servidor. Como em um sistema de arquivos normal, você pode criar, renomear, e exclua arquivos e pastas.

USANDO A LINHA DE COMANDO PARA INTERAGIR COM O SISTEMA DE ARQUIVOS HADOOP

Vamos primeiro recuperar a lista atualmente presente de diretórios e arquivos na raiz do Hadoop pasta usando a linha de comando. Digite o comando hadoop fs –ls / no PuTTY para Alcançar isso.

Certifique-se de ligar sua máquina virtual com o Hortonworks Sandbox antes tentando uma conexão. No PuTTY você deve então conectar-se a 127.0.0.1:2222, como mostrado antes na figura 5.6.

A saída do comando Hadoop é mostrada na figura 5.8. Você também pode adicionar argumentos como hadoop fs –ls –R / para obter uma lista recursiva de todos os arquivos e subdiretórios.

```
[root@sandbox ~]# hadoop fs -ls /
Found 20 items
drwxrwxrwx   - admin  hadoop          0 2015-07-14 14:54 /LoanStats3c.cs
-rw-r--r--   1 root   hadoop  120834552 2015-07-14 14:47 /LoanStats3c.csv
drwxrwxrwx   - yarn   hadoop          0 2015-07-15 13:32 /app-logs
drwxr-xr-x   - hdfs   hdfs            0 2015-06-05 09:19 /apps
drwxr-xr-x   - admin  hadoop          0 2015-07-13 06:47 /book
drwxr-xr-x   - root   hadoop          0 2015-07-17 10:24 /chapter5
-rwxr-xr-x   1 hdfs   hadoop       4240 2015-07-14 19:32 /cout.json
```

Figura 5.8 Saída do comando list do Hadoop: hadoop fs –ls /. A pasta raiz do Hadoop está listada.

Agora criaremos um novo diretório “chapter5” no hdfs para trabalhar durante este capítulo. Os comandos a seguir criarão o novo diretório e darão acesso a todos a pasta:

```
sudo -u hdfs hadoop fs -mkdir /chapter5
sudo -u hdfs hadoop fs –chmod 777 /chapter5
```

Você provavelmente notou um padrão aqui. Os comandos do Hadoop são muito semelhantes aos nossos comandos do sistema de arquivos local (estilo POSIX), mas comece com Hadoop fs e tenha um traço - antes de cada comando. A Tabela 5.1 fornece uma visão geral dos comandos populares do sistema de arquivos no Hadoop e seus equivalentes de comando do sistema de arquivos local.

Tabela 5.1 Lista de comandos comuns do sistema de arquivos Hadoop

Meta	Comando do sistema de arquivos Hadoop	Comando do sistema de arquivos local
Obtenha uma lista de arquivos e diretórios de um diretório	hadoop fs –ls URI	é URI
Crie um diretório	hadoop fs –mkdir URI	URI mkdir
Remover um diretório	hadoop fs –rm –r URI	rm –r URI

Tabela 5.1 Lista de comandos comuns do sistema de arquivos Hadoop

Meta	Comando do sistema de arquivos Hadoop	Comando do sistema de arquivos local
Alterar a permissão dos arquivos	hadoop fs –chmod URI do MODO	chmod MODO URI
Mover ou renomear arquivo	hadoop fs –mv OLDURI NEWURI	mv OLDURI NEWURI

Existem dois comandos especiais que você usará com frequência. São eles: ÿ

Fazer upload de arquivos do sistema de arquivos local para o sistema de arquivos distribuído (hadoop fs -coloque LOCAIS REMOTOS).

ÿ Faça download de um arquivo do sistema de arquivos distribuído para o sistema de arquivos local (hadoop –obter REMOTEUR).

Vamos esclarecer isso com um exemplo. Suponha que você tenha um arquivo .CSV no Linux virtual máquina a partir da qual você se conecta ao cluster Linux Hadoop. Você deseja copiar o

Arquivo .CSV da sua máquina virtual Linux para o cluster hdfs. Use o comando hadoop –put mycsv.csv /data.

Usando PuTTY, podemos iniciar uma sessão Python no Horton Sandbox para recuperar nosso dados usando um script Python. Emita o comando “pyspark” na linha de comando para iniciar a sessão. Se tudo estiver bem você deverá ver a tela de boas-vindas mostrada na figura 5.9.

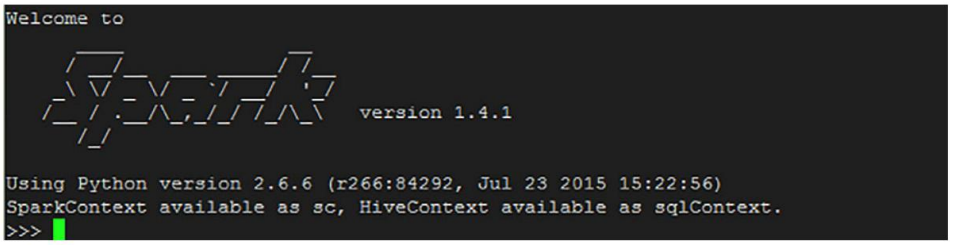


Figura 5.9 A tela de boas-vindas do Spark para uso interativo com Python

Agora usamos o código Python para buscar os dados para nós, conforme mostrado na listagem a seguir.

Listagem 5.1 Extração de dados de empréstimo do Lending Club

```
solicitações de importação
importar arquivo zip
importar StringIO
fonte = solicitações.get("https://resources.lendingclub.com/
        LoanStats3d.csv.zip", verify=False) stringio =
StringIO.StringIO(source.content) descompactado = zipfile.ZipFile(stringio)
```

Baixa dados de Clube de empréstimos. Isto é https, então deve ser verificado, mas não vamos nos preocupar (verify=False).

Cria arquivo virtual.

Descompacta dados.

Baixamos o arquivo “LoanStats3d.csv.zip” do site do Lending Club em https://
resources.lendingclub.com/LoanStats3d.csv.zip e descompacte-o. Usamos métodos de
os pacotes requests, zipfile e stringio Python para baixar os dados, respectivamente,
crie um arquivo virtual e descompacte-o. Este é apenas um único arquivo; se você quiser todos os dados deles, você
poderia criar um loop, mas para fins de demonstração isso servirá. Como mencionamos
antes, uma parte importante deste estudo de caso será a preparação de dados com big data
tecnologias. Antes de podermos fazer isso, porém, precisamos colocá-lo no sistema de arquivos Hadoop. PyWebHdfs
é um pacote que permite interagir com o sistema de arquivos Hadoop
do Python. Ele traduz e passa seus comandos para chamadas restantes para o webhdfs
interface. Isso é útil porque você pode usar sua linguagem de script favorita para automatizar tarefas, conforme
mostrado na listagem a seguir.

Listagem 5.2 Armazenando dados no Hadoop

Armazena-o localmente porque precisamos transferi-lo para o sistema de arquivos Hadoop.

Faz limpeza preliminar de dados usando Pandas: remove linha superior e 2 linhas inferiores porque são inúteis. Abrir o arquivo original mostrará isso.

```
importar pandas como pd de
pywebhdfs.webhdfs importar PyWebHdfsClient
subselection_csv = pd.read_csv(descompactado.open('LoanStats3d.csv'),
        skiprows=1,skipfooter=2,engine='python')
armazenado_csv = subselection_csv.to_csv('./stored_csv.csv') hdfs =
PyWebHdfsClient(user_name="hdfs",port=50070,host="sandbox")
hdfs.make_dir('capítulo5')
com open('./stored_csv.csv') como file_data:
        hdfs.create_file('chapter5/LoanStats3d.csv',file_data,
    substituir=Verdadeiro)
```

Cria a pasta "chapter5" no sistema de arquivos Hadoop.

Conecta-se ao Hadoop Sandbox.

Cria arquivo .csv em Sistema de arquivos Hadoop.

Abre csv armazenado localmente.

Já havíamos baixado e descompactado o arquivo da listagem 5.1; agora na listagem 5.2 nós
fez uma subseleção dos dados usando Pandas e os armazenou localmente. Então criamos um
diretório no Hadoop e transferiu o arquivo local para o Hadoop. Os dados baixados
está no formato .CSV e por ser bastante pequeno, podemos usar a biblioteca Pandas para
remova a primeira linha e as duas últimas linhas do arquivo. Eles contêm comentários e serão
apenas torna o trabalho com esse arquivo complicado em um ambiente Hadoop. A primeira linha
do nosso código importa o pacote Pandas, enquanto a segunda linha analisa o arquivo em
memória e remove a primeira e as duas últimas linhas de dados. A terceira linha de código salva o
dados para o sistema de arquivos local para uso posterior e fácil inspeção.

Antes de prosseguir, podemos verificar nosso arquivo usando a seguinte linha de código:

```
imprimir hdfs.get_file_dir_status('chapter5/LoanStats3d.csv')
```

O console PySpark deve nos dizer que nosso arquivo está seguro e funcionando bem no sistema Hadoop, como
mostrado na figura 5.10.

```
>>> print hdfs.get_file_dir_status('chapter5/LoanStats3d.csv')#A
{u'FileStatus': {u'group': u'hdfs', u'permission': u'755', u'blockSize': 1342177
28, u'accessTime': 1449236321223, u'pathSuffix': u'', u'modificationTime': 14492
36321965, u'replication': 3, u'length': 120997124, u'childrenNum': 0, u'owner':
u'hdfs', u'storagePolicy': 0, u'type': u'FILE', u'fileId': 17520}}
```

Figura 5.10 Recuperar o status do arquivo no Hadoop por meio do console PySpark

Com o arquivo pronto e esperando por nós no Hadoop, podemos prosseguir para a preparação dos dados usando o Spark, porque não é limpo o suficiente para ser armazenado diretamente no Hive.

5.2.3 Etapa 3: Preparação dos dados

Agora que baixamos os dados para análise, usaremos o Spark para limpar os dados antes de armazená-lo no Hive.

PREPARAÇÃO DE DADOS NO SPARK

A limpeza de dados costuma ser um exercício interativo, porque você identifica um problema e corrige o problema. problema, e você provavelmente fará isso algumas vezes antes de obter resultados limpos e nítidos dados. Um exemplo de dados sujos seria uma string como “UsA”, que é incorretamente capitalizado. Neste ponto, não trabalhamos mais em jobs.py, mas usamos o comando PySpark interface de linha para interagir diretamente com o Spark.

O Spark é adequado para esse tipo de análise interativa porque não precisa salve os dados após cada etapa e tenha um modelo muito melhor que o Hadoop para compartilhamento dados entre servidores (uma espécie de memória distribuída).

A transformação consiste em quatro partes:

1 Inicie o PySpark (ainda deve estar aberto na seção 5.2.2) e carregue o Spark e contexto do Hive.

2 Leia e analise o arquivo .CSV .

3 Divida a linha do cabeçalho dos dados.

4 Limpe os dados.

Ok, vamos aos negócios. A listagem a seguir mostra a implementação do código no Console PySpark.

Listagem 5.3 Conectando-se ao Apache Spark

Importa contexto do Spark --> não é necessário ao trabalhar diretamente no PySpark.

Cria o contexto do Hive.

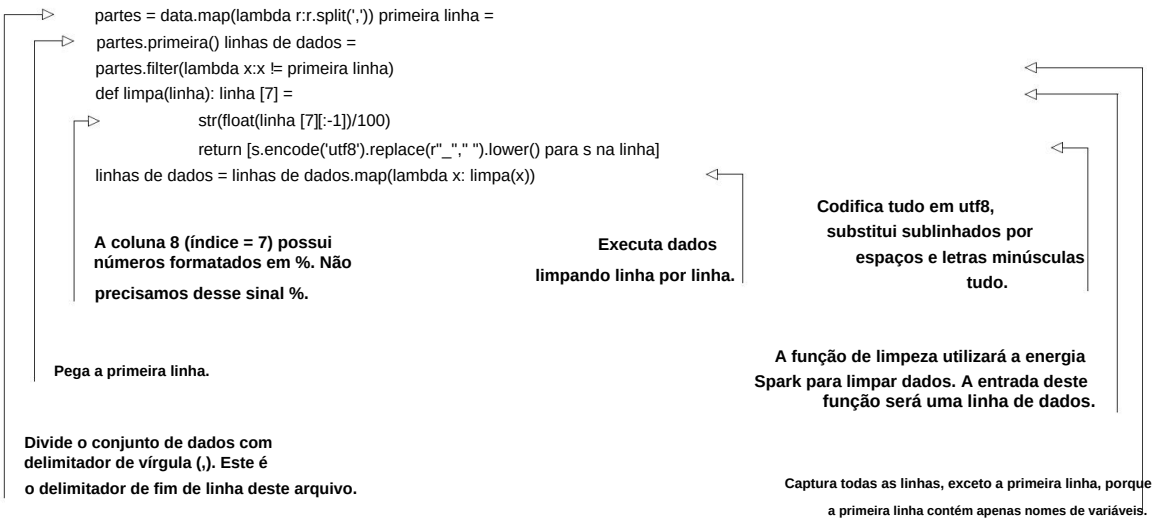
Importa o contexto do Hive.

Na sessão PySpark, o contexto Spark está presente automaticamente. Em outros casos (notebook Zeppelin), você precisará criá-lo explicitamente.

```
de pyspark importar SparkContext de pyspark.sql
importar HiveContext #sc = SparkContext()

sqlContext = HiveContext(sc)
dados = sc.textFile("/chapter5/LoanStats3d.csv")
```

Carrega no conjunto de dados de Diretório Hadoop.



Vamos nos aprofundar um pouco mais nos detalhes de cada etapa.

Etapa 1: iniciar o Spark no modo interativo e carregar o contexto

A importação de contexto do Spark não é necessária no console do PySpark porque um contexto é prontamente disponível como variável sc. Você deve ter notado que isso também é mencionado quando abrindo o PySpark; verifique a figura 5.9 caso você a tenha esquecido. Em seguida, carregamos um Contexto do Hive para nos permitir trabalhar interativamente com o Hive. Se você trabalha interativamente com Spark, os contextos Spark e Hive são carregados automaticamente, mas se você quiser use-o no modo em lote, você precisa carregá-lo manualmente. Para enviar o código em lote você usaria o comando spark-submit filename.py na linha de comando do Horton Sandbox.

```
de pyspark importar SparkContext de pyspark.sql
importar HiveContext sc = SparkContext()

sqlContext = HiveContext(sc)
```

Com o ambiente configurado, estamos prontos para começar a analisar o arquivo .CSV .

Etapa 2: Ler e analisar o arquivo .CSV

Em seguida, lemos o arquivo do sistema de arquivos Hadoop e o dividimos em cada vírgula que encontro. Em nosso código, a primeira linha lê o arquivo .CSV do sistema de arquivos Hadoop. A segunda linha divide todas as linhas quando encontra uma vírgula. Nosso analisador .CSV é ingênuo por design porque estamos aprendendo sobre o Spark, mas você também pode usar o .CSV pacote para ajudá-lo a analisar uma linha mais corretamente.

```
dados = sc.textFile("/chapter5/LoanStats3d.csv") partes = data.map(lambda
r:r.split(','))
```

Observe como isso é semelhante a uma abordagem de programação funcional. Para aqueles que nunca o encontraram, você pode ler ingenuamente lambda r:r.split(',') como “para cada entrada r (uma linha neste caso), divida esta entrada r quando encontrar uma vírgula”. Como neste caso, “para cada entrada” significa “para cada linha”, mas você também pode ler como “dividir cada linha por uma vírgula”. Essa sintaxe funcional é uma das minhas características favoritas do Spark.

Etapa 3: divida a linha do cabeçalho dos dados

Para separar o cabeçalho dos dados, lemos a primeira linha e retemos todas as linhas que não são semelhantes à linha do cabeçalho:

```
primeira linha = parts.first() linhas de
dados = parts.filter (lambda x:x! = primeira linha)
```

Seguindo as melhores práticas em big data, não precisaríamos fazer esta etapa porque a primeira linha já estaria armazenada em um arquivo separado. Na realidade, os arquivos .CSV geralmente contêm uma linha de cabeçalho e você precisará realizar uma operação semelhante antes de começar a limpar os dados.

Etapa 4: Limpe os dados

Nesta etapa realizamos uma limpeza básica para melhorar a qualidade dos dados. Isso nos permite construir um relatório melhor.

Após a segunda etapa, nossos dados consistem em arrays. Trataremos cada entrada para uma função lambda como um array agora e retornaremos um array. Para facilitar essa tarefa, construímos uma função auxiliar que limpa. Nossa limpeza consiste em reformatar uma entrada como “10,4%” para 0,104 e codificar cada string como utf-8, bem como substituir sublinhados por espaços e colocar todas as strings em minúsculas. A segunda linha de código chama nossa função auxiliar para cada linha do array.

```
def limpa(linha): linha [7] =
                str(float(linha [7][:-1])/100) return [s.encode('utf8').replace(r"_",'
                '). lower() para s na linha]
linhas de dados = linhas de dados.map(lambda x: limpa(x))
```

Nossos dados estão agora preparados para o relatório, por isso precisamos disponibilizá-los para nossas ferramentas de relatório. O Hive é adequado para isso, porque muitas ferramentas de relatórios podem se conectar a ele. Vejamos como fazer isso.

SALVAR OS DADOS NO HIVE

Para armazenar dados no Hive, precisamos concluir duas etapas:

1 Crie e registre metadados.

2 Execute instruções SQL para salvar dados no Hive.

Nesta seção, executaremos mais uma vez o próximo trecho de código em nosso amado shell PySpark, conforme mostrado na listagem a seguir.

Listagem 5.4 Armazenando dados no Hive (completo)

Cria metadados: a função Spark SQL StructField representa um campo em um StructType. O objeto StructField é composto por três campos: nome (uma string), dataType (um DataType) e "anulável" (um booleano). O campo de nome é o nome de um StructField. O campo de dataType especifica o tipo de dados de um StructField. O campo nullable especifica se os valores de um StructField podem conter valores None.

Importa tipos de dados SQL.

```
de pyspark.sql.types importar * campos =
[StructField(field_name,StringType(),True) para field_name em
        primeira linha]
esquema = StructType(campos)
esquemaLoans = sqlContext.createDataFrame(linhas de dados, esquema)
esquemaLoans.registerTempTable("empréstimos")
```

Cria quadro de dados a partir de dados (linhas de dados) e esquema de dados (esquema).

Registra-o como uma tabela chamada empréstimos.

A função StructType cria o esquema de dados. Um objeto StructType requer uma lista de StructFields como entrada.

```
sqlContext.sql("eliminar tabela se existir LoansByTitle")
sql = "'criar tabela LoansByTitle armazenada como parquet como título selecionado,
        contagem (1) como número do grupo de empréstimos por título, ordem por número desc"' sqlContext.sql(sql)

sqlContext.sql('eliminar tabela se existir raw')
sql = "'criar tabela bruta armazenada como parquet como select title,
        emp_title,grade,home_ownership,int_rate,recoveries,
        taxa_de_recuperação_de_cobrança,amnt_do_empréstimo,prazo dos empréstimos"'
```

Elimina a tabela (caso ela já exista) e armazena um subconjunto de dados brutos no Hive.

Descarta a tabela (caso já exista), resume e armazena no Hive. LoansByTitle representa a soma dos empréstimos por cargo.

Vamos nos aprofundar em cada etapa para obter mais esclarecimentos.

Passo 1: Criar e registrar metadados

Muitas pessoas preferem usar SQL quando trabalham com dados. Isto também é possível com Fagulha. Você pode até ler e armazenar dados diretamente no Hive, como faremos. Antes que você possa fazer entretanto, você precisará criar metadados que contenham um nome de coluna e um tipo de coluna para cada coluna.

A primeira linha do código são as importações. A segunda linha analisa o nome do campo e o tipo de campo e especifica se um campo é obrigatório. O StructType representa linhas como um matriz de campos de estrutura. Em seguida, você o coloca em um dataframe registrado como uma tabela (temporária) no Hive.

```
de pyspark.sql.types importar * campos =
[StructField(field_name,StringType(),True) para field_name na primeira linha]
esquema = StructType(campos)
esquemaLoans = sqlContext.createDataFrame(linhas de dados, esquema)
esquemaLoans.registerTempTable("empréstimos")
```

Com os metadados prontos, agora podemos inserir os dados no Hive.

Etapa 2: executar consultas e armazenar tabela no Hive

Agora estamos prontos para usar um dialeto SQL em nossos dados. Primeiro faremos uma tabela resumo que conta o número de empréstimos por finalidade. Então armazenamos um subconjunto do limpo dados brutos no Hive para visualização no Qlik.

Executar comandos do tipo SQL é tão fácil quanto passar uma string que contém o comando SQL para a função sqlContext.sql . Observe que não estamos escrevendo SQL puro porque estamos nos comunicando diretamente com o Hive. O Hive tem seu próprio dialeto SQL chamado ColmeiaQL. Em nosso SQL, por exemplo, dizemos imediatamente para armazenar os dados como um Parquet arquivo. Parquet é um formato popular de arquivo de big data.

```
sqlContext.sql("descartar tabela se existir LoansByTitle") sql = '''criar tabela
LoansByTitle armazenada como parquet como título selecionado,
        contar (1) como número do grupo de empréstimos por título, ordenar por número desc'''
sqlContext.sql(sql)
```

```
sqlContext.sql('descartar tabela se existir raw') sql = '''criar
tabela raw armazenada como parquet como select title,
        emp_title,grade,home_ownership,int_rate,recoveries,collection_recovery_f
        ee,loan_amnt,prazo dos empréstimos'''
sqlContext.sql(sql)
```

Com os dados armazenados no Hive, podemos conectar nossas ferramentas de visualização a ele.

5.2.4 Etapa 4: Exploração de dados e Etapa 6: Construção de relatórios

Construiremos um relatório interativo com Qlik Sense para mostrar ao nosso gerente. Qlik Sense pode ser baixado em http://www.qlik.com/try-or-buy/download-qlik-sense após assinando seu site. Quando o download começar você será redirecionado para um página contendo vários vídeos informativos sobre como instalar e trabalhar com Qlik Senso. É recomendado assisti-los primeiro.

Usamos o conector Hive ODBC para ler dados do Hive e disponibilizá-los para Qlik. Um tutorial sobre como instalar conectores ODBC no Qlik está disponível. Para os principais sistemas operacionais, isso pode ser encontrado em http://hortonworks.com/hdp/addons/.

NOTA No Windows, isso pode não funcionar imediatamente. Depois de instalar o ODBC, certifique-se de verificar o gerenciador ODBC do Windows (CTRL+F e procure para ODBC). No gerenciador, vá em “System-DSN” e selecione “Sample Hive Hortonworks DSN”. Certifique-se de que suas configurações estejam corretas (conforme mostrado na figura 5.11) ou o Qlik não se conectará ao Hortonworks Sandbox.

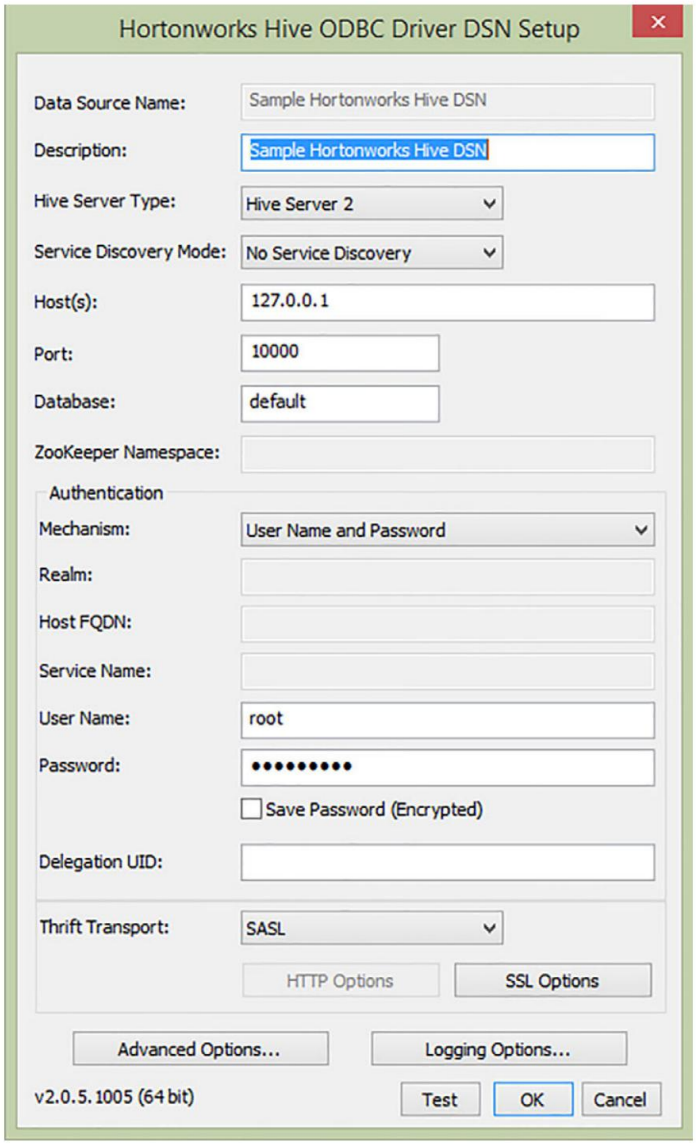


Figura 5.11 Configuração ODBC do Windows Hortonworks

Esperamos que você não tenha esquecido sua senha do Sandbox; como você pode ver na figura 5.11, você
preciso disso novamente.

Agora abra o Qlik Sense. Se instalado no Windows você deveria ter obtido a opção
para colocar um atalho para o .exe em sua área de trabalho. Qlik não é freeware; é um comercial

produto com versão isca para clientes individuais, mas será suficiente por enquanto. No último capítulo criaremos um painel usando bibliotecas JavaScript gratuitas.

O Qlik pode levar os dados diretamente para a memória ou fazer uma chamada sempre para o Hive. Escolhemos o primeiro método porque funciona mais rápido.

Esta parte tem três etapas:

1 Carregue dados dentro do Qlik com uma conexão ODBC .

2 Crie o relatório.

3 Explore os dados.

Vamos começar com a primeira etapa, carregar dados no Qlik.

Passo 1: Carregar dados no

Qlik Ao iniciar o Qlik Sense ele irá mostrar uma tela de boas-vindas com os relatórios existentes (chamados de apps), conforme mostrado na figura 5.12.

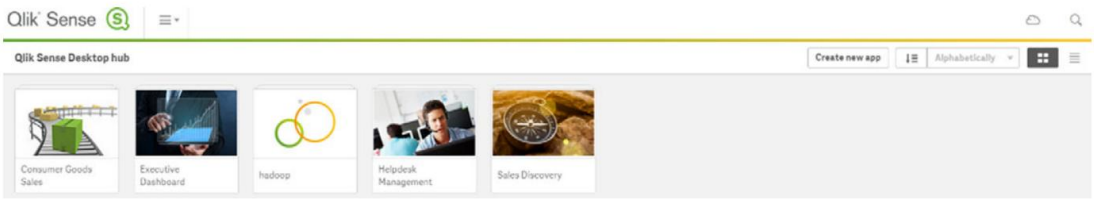


Figura 5.12 A tela de boas-vindas do Qlik Sense

Para iniciar um novo aplicativo, clique no botão *Criar novo aplicativo* à direita da tela, conforme mostrado na figura 5.13. Isso abre uma nova caixa de diálogo. Digite “capítulo 5” como o novo nome do nosso aplicativo.

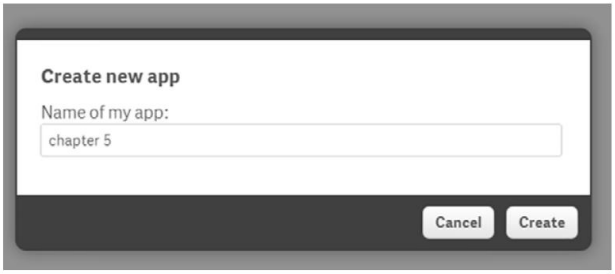


Figura 5.13 A caixa de mensagem Criar novo aplicativo

Uma caixa de confirmação aparece (figura 5.14) se o aplicativo for criado com sucesso.

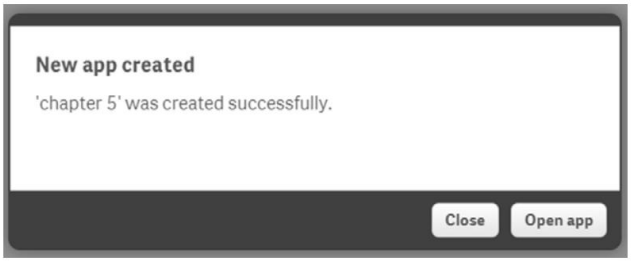


Figura 5.14 Uma caixa confirma que o aplicativo foi criado com sucesso.

Clique no botão Abrir aplicativo e uma nova tela solicitará que você adicione dados ao aplicação (figura 5.15).

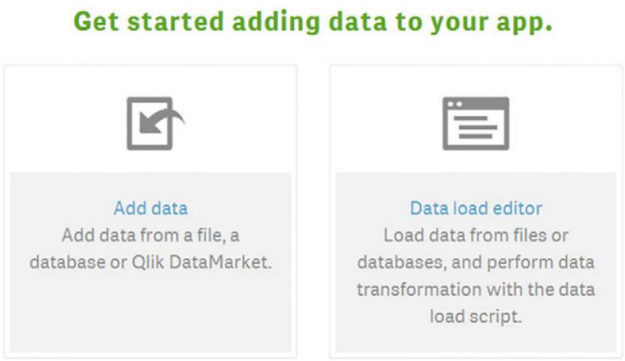


Figura 5.15 Uma tela para iniciar a adição de dados aparece quando você abre um novo aplicativo.

Clique no botão Adicionar dados e escolha ODBC como fonte de dados (figura 5.16).

Na próxima tela (figura 5.17) selecione User DSN, Hortonworks, e especifique o root como nome de usuário e hadoop como senha (ou a nova que você forneceu ao fazer login na Sandbox pela primeira vez).

> **NOTA** A opção Hortonworks não aparece por padrão. Você precisa instale o conector ODBC HDP 2.3 para que esta opção apareça (conforme indicado antes). Se você não conseguiu instalá-lo neste momento, instruções claras para isso podem ser encontradas em https://blogs.perficient.com/multi-shoring/blog/2015/09/29/how-to-connect-hortonworks-hive-from-qlikview-with-odbc-driver/.

Clique na seta apontando para a direita para ir para a próxima tela.

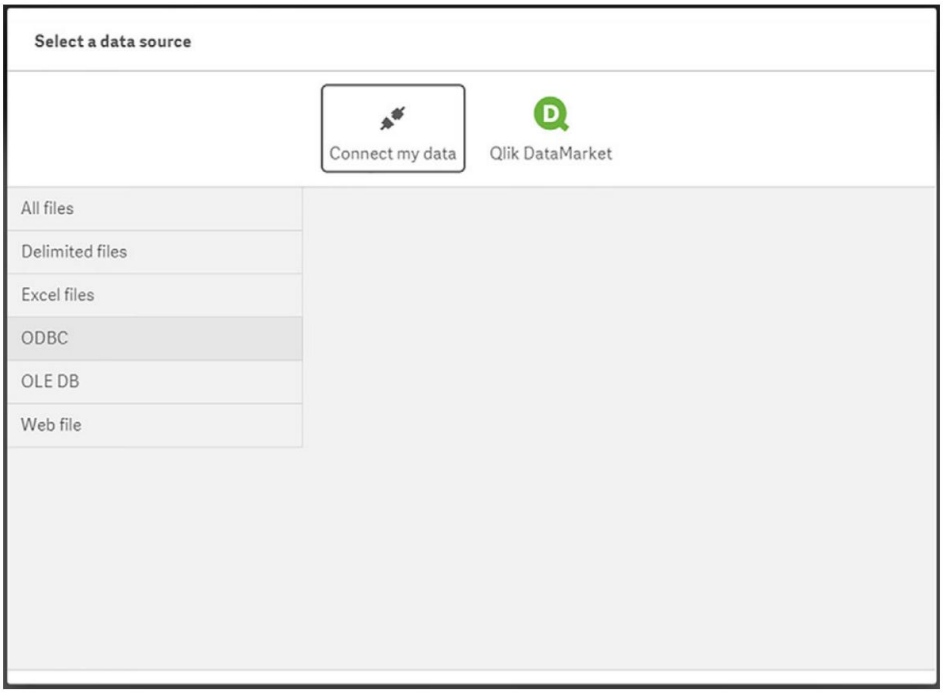


Figura 5.16 Escolha ODBC como fonte de dados na tela Selecionar uma fonte de dados

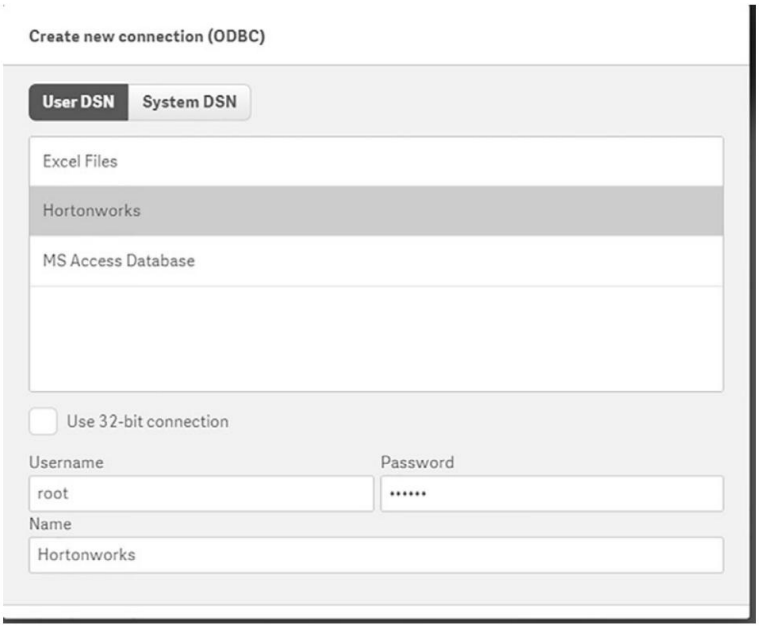


Figura 5.17 Escolha Hortonworks no DSN do usuário e especifique o nome de usuário e a senha.

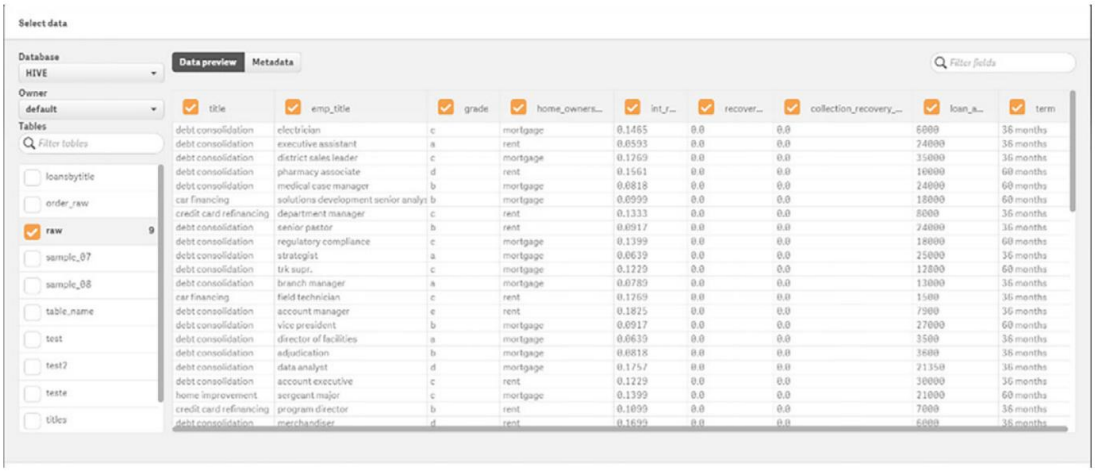


Figura 5.18 Visão geral da coluna de dados brutos da interface Hive

Escolha os dados do Hive e use como usuário padrão na próxima tela (figura 5.18). Selecione bruto como Tabelas para selecionar e selecionar todas as colunas para importação; em seguida, clique no botão Carregar e Concluir para concluir esta etapa.

Após esta etapa, levará alguns segundos para carregar os dados no Qlik (figura 5.19).

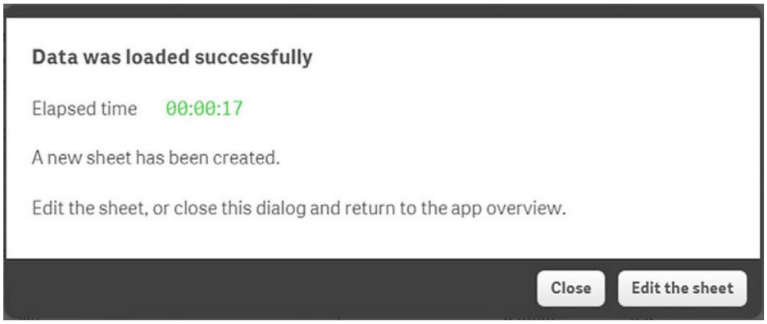


Figura 5.19 Uma confirmação de que os dados estão carregados no Qlik

Etapa 2: Criar o relatório

Escolha Editar a planilha para começar a construir o relatório. Isto irá adicionar o editor de relatórios (figura 5.20).

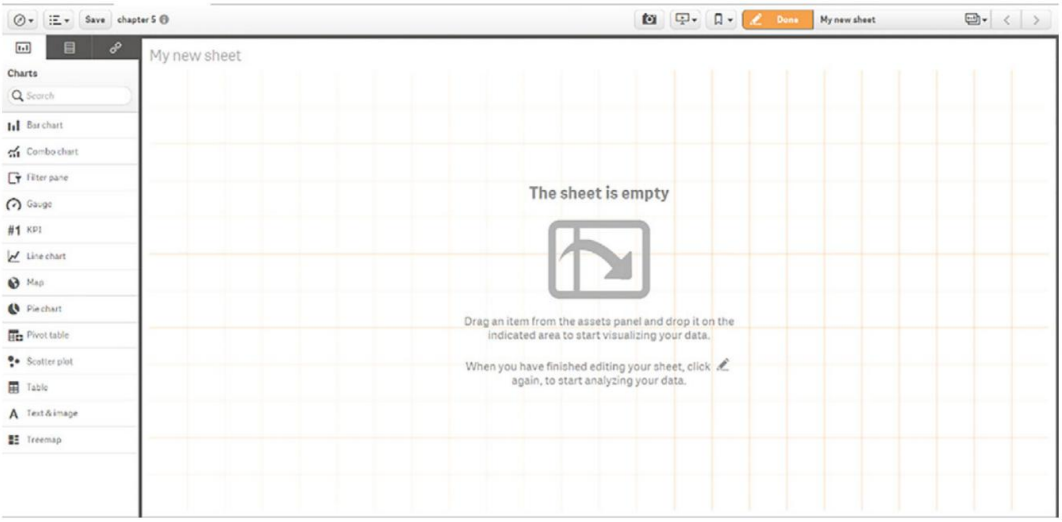


Figura 5.20 Uma tela do editor para relatórios é aberta

Subetapa 1: Adicionando um filtro de seleção ao relatório A primeira coisa que adicionaremos ao relatório é uma caixa de seleção que mostra por que cada pessoa deseja um empréstimo. Para conseguir isso, solte a medida do título do painel de ativos esquerdo no painel do relatório e atribua-lhe um tamanho e uma posição confortáveis (figura 5.21). Clique na tabela Campos para poder arrastar e soltar campos.

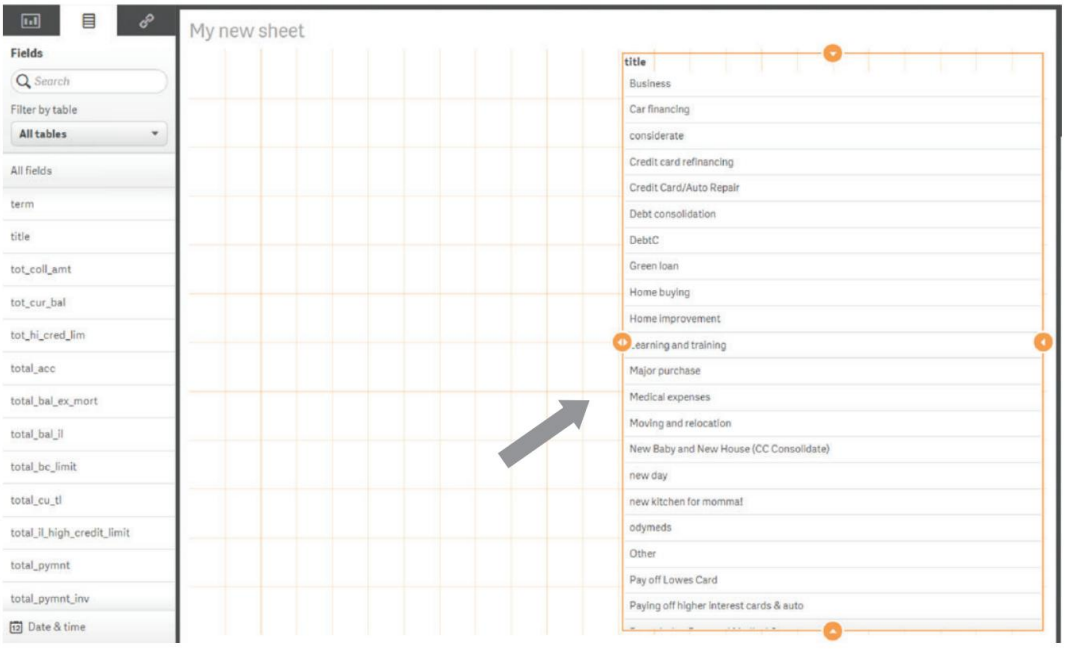


Figura 5.21 Arraste o título do painel esquerdo Campos para o painel do relatório.

Subetapa 2: Adicionando um KPI ao relatório Um gráfico de KPI mostra um número agregado para a população total selecionada. Números como a taxa média de juros e o número total de clientes são mostrados neste gráfico (figura 5.22).

Figura 5.22 Um exemplo de gráfico de KPI

Adicionar um KPI a um relatório leva quatro etapas, conforme listado abaixo e mostrado na figura 5.23.

1 *Escolha um* gráfico—Escolha KPI como gráfico e coloque-o na tela de relatório; redimensione e posicione ao seu gosto.

2 *Adicionar uma medida — Clique* no botão adicionar medida dentro do gráfico e selecione int_rate.

3 *Escolha um* método de agregação — Avg(int_rate).

4 *Formate o gráfico – No* painel direito, preencha a taxa de juros média como Rótulo.

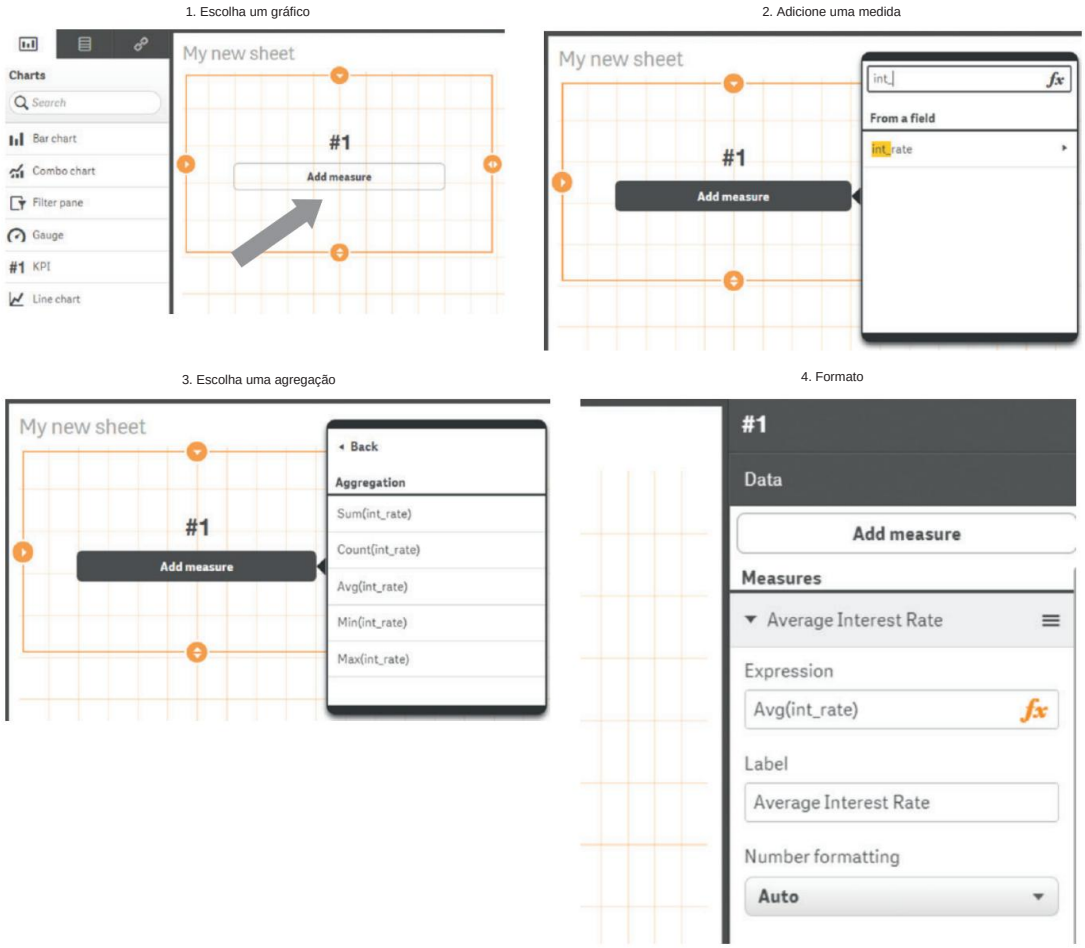


Figura 5.23 As quatro etapas para adicionar um gráfico de KPI a um relatório Qlik

No total, adicionaremos quatro gráficos de KPI ao nosso relatório, portanto, você precisará repetir essas etapas para os seguintes KPIs:

ÿ Taxa média de juros ÿ Valor
total do empréstimo

ÿ Valor médio do empréstimo ÿ
Total de recuperações

Subetapa 3: Adicionando gráficos de barras ao nosso relatório A seguir adicionaremos quatro gráficos de barras ao relatório. Eles mostrarão os diferentes números para cada grau de risco. Um gráfico de barras explicará a taxa de juro média por grupo de risco e outro mostrar-nos-á o montante total do empréstimo por grupo de risco (figura 5.24).

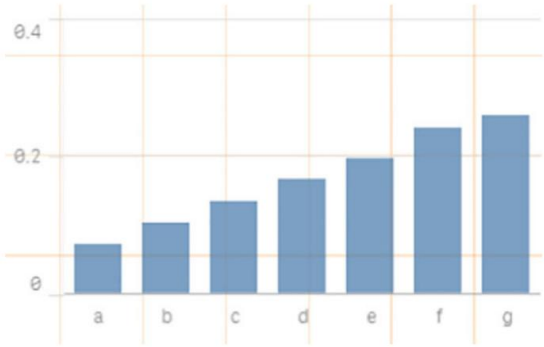


Figura 5.24 Um exemplo de gráfico de barras

Adicionar um gráfico de barras a um relatório requer cinco etapas, conforme listado abaixo e mostrado na figura 5.25.

1 *Escolha um* gráfico—Escolha o gráfico de barras como gráfico e coloque-o na tela do relatório; redimensione e posicione ao seu gosto.

2 *Adicionar uma medida — Clique* no botão Adicionar medida dentro do gráfico e selecione taxa_int.

3 *Escolha um* método de agregação — Avg(int_rate).

4 *Adicionar uma dimensão — Clique em* Adicionar dimensão e escolha a nota como a dimensão.

5 *Formate o gráfico – No* painel direito, preencha a taxa de juros média como Rótulo.

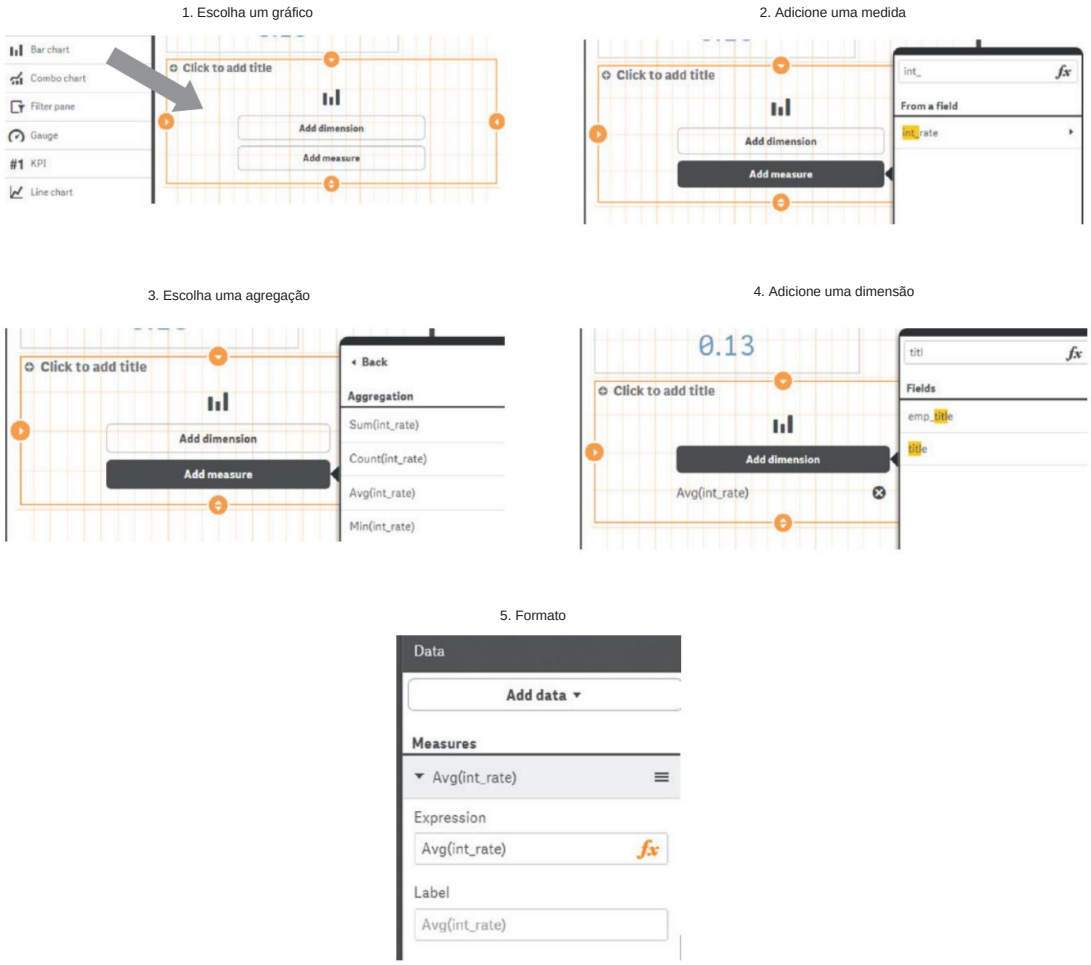


Figura 5.25 Adicionar um gráfico de barras requer cinco etapas.

Repita este procedimento para as seguintes combinações de dimensões e medidas:

ÿ Taxa média de juros por categoria ÿ
Valor médio do empréstimo por categoria
ÿ Valor total do empréstimo por
categoria ÿ Total de recuperações por categoria

Subetapa 4: Adicionar uma tabela cruzada ao relatório Suponha que você queira saber a taxa de juros média paga pelos diretores do grupo de risco C. Neste caso você deseja obter uma medida (taxa de juros) para uma combinação de duas dimensões (cargo e grau de risco). Isto pode ser conseguido com uma tabela dinâmica como na figura 5.26.

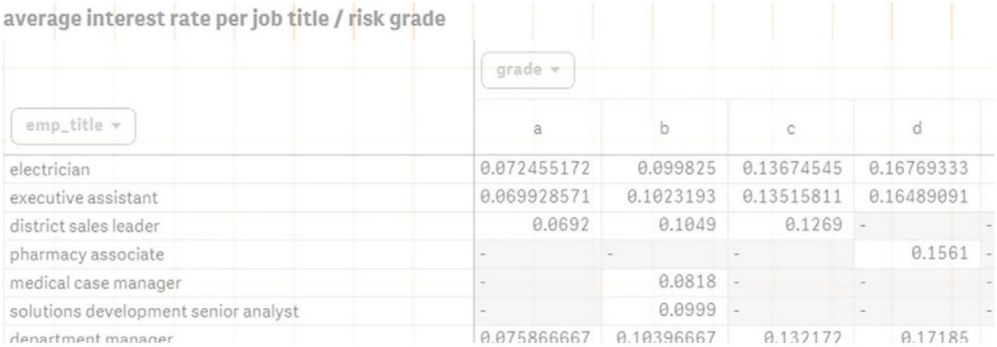


Figura 5.26 Um exemplo de tabela dinâmica, mostrando a taxa média de juros paga por combinação de cargo/grau de risco

Adicionar uma tabela dinâmica a um relatório requer seis etapas, conforme listado abaixo e mostrado na figura 5.27.

1 *Escolha um* gráfico - escolha a tabela dinâmica como gráfico e coloque-a no relatório tela; redimensione e posicione ao seu gosto.

2 *Adicionar uma medida — Clique* no botão Adicionar medida dentro do gráfico e selecione taxa_int.

3 *Escolha um* método de agregação — Avg(int_rate).

4 *Adicionar uma dimensão de linha: clique em* Adicionar dimensão e escolha emp_title como dimensão.

5 *Adicionar uma dimensão de coluna: clique em* Adicionar dados, escolha a coluna e selecione a nota.

6 *Formate o gráfico – No* painel direito, preencha a taxa de juros média como Rótulo.

Figura 5.27 Adicionar uma tabela dinâmica requer seis etapas.

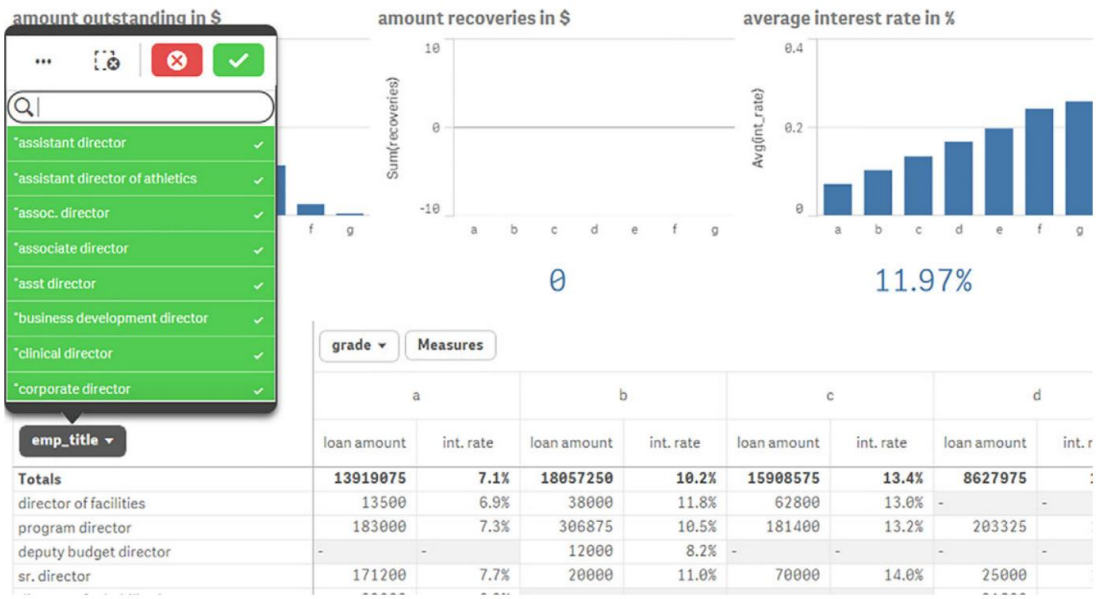
Figura 5.28 O resultado final no modo de edição

Após redimensionar e reposicionar, você deverá obter um resultado semelhante ao da figura 5.28. Clique no botão Concluído à esquerda e você estará pronto para explorar os dados.

Etapa 3: explore os dados

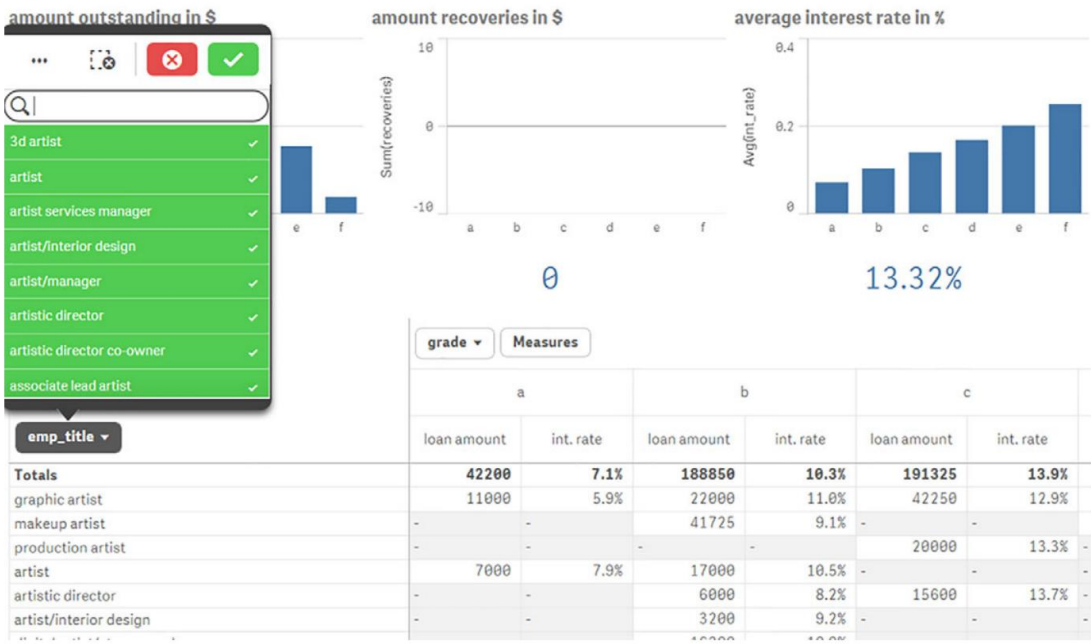
O resultado é um gráfico interativo que se atualiza com base nas seleções feitas. Por que você não tenta buscar as informações dos diretores e compará-las com artistas? Para conseguir isso, clique em emp_title na tabela dinâmica e digite director no campo de pesquisa. O resultado se parece com a figura 5.29. Da mesma forma, podemos olhar para os artistas, conforme mostra a figura 5.30. Outro insight interessante vem da comparação da classificação para fins de compra de casa com fins de consolidação de dívidas.

Finalmente conseguimos: criamos o relatório que nosso gerente deseja e, no processo, abriu a porta para que outras pessoas criassem seus próprios relatórios usando esses dados. Um próximo passo interessante para você refletir seria usar esta configuração para encontrar essas pessoas provavelmente não pagarão suas dívidas. Para isso, você pode usar os recursos de aprendizado de máquina do Spark conduzidos por algoritmos online como os demonstrados no capítulo 4.



	a		b		c		d	
	loan amount	int. rate	loan amount	int. rate	loan amount	int. rate	loan amount	int. r
Totals	**13919075**	**7.1%**	**18057250**	**10.2%**	**15908575**	**13.4%**	**8627975**	
director of facilities	13500	6.9%	38000	11.8%	62800	13.0%	-	-
program director	183000	7.3%	306875	10.5%	181400	13.2%	203325	
deputy budget director	-	-	12000	8.2%	-	-	-	-
sr. director	171200	7.7%	20000	11.0%	70000	14.0%	25000	

Figura 5.29 Quando selecionamos diretores, podemos ver que eles pagam uma taxa média de 11,97% por um empréstimo.



	a		b		c	
	loan amount	int. rate	loan amount	int. rate	loan amount	int. rate
Totals	**42200**	**7.1%**	**188850**	**10.3%**	**191325**	**13.9%**
graphic artist	11000	5.9%	22000	11.0%	42250	12.9%
makeup artist	-	-	41725	9.1%	-	-
production artist	-	-	-	-	20000	13.3%
artist	7000	7.9%	17000	10.5%	-	-
artistic director	-	-	6000	8.2%	15600	13.7%
artist/interior design	-	-	3200	9.2%	-	-

Figura 5.30 Quando selecionamos artistas, vemos que eles pagam uma taxa média de juros de 13,32% por um empréstimo.

Neste capítulo, tivemos uma introdução prática às estruturas Hadoop e Spark.
Cobrimos muito assunto, mas seja honesto, Python torna o trabalho com tecnologias de big data extremamente fácil. No próximo capítulo iremos nos aprofundar no mundo dos bancos de dados NoSQL e entrar em contato com mais tecnologias de big data.

5.3 Resumo

Neste capítulo você aprendeu que

- ÿ Hadoop é uma estrutura que permite armazenar arquivos e distribuir cálculos entre vários computadores.
- ÿ O Hadoop oculta para você todas as complexidades de trabalhar com um cluster de computadores.
- ÿ Um ecossistema de aplicativos envolve o Hadoop e o Spark, desde bancos de dados até controle de acesso.
- ÿ O Spark adiciona uma estrutura de memória compartilhada ao Hadoop Framework que é melhor adequado para trabalho de ciência de dados.
- ÿ No capítulo de estudo de caso, usamos PySpark (uma biblioteca Python) para comunicar com Hive e Spark do Python. Usamos a biblioteca Python pywebhdfs para funcionar com a biblioteca Hadoop, mas você também pode usar a linha de comando do sistema operacional .
- ÿ É fácil conectar uma ferramenta de BI como o Qlik ao Hadoop.