第 5 章
数 据 分 析

学习目标

- 了解数据分析，能够说出数据分析的常用方法；
- 掌握部署 Hive，能够独立完成本地模式部署 Hive 的操作；
- 了解数据仓库的概念，能够说出数据仓库的特点；
- 熟悉数据仓库的设计，能够说出数据仓库分层的作用；
- 掌握数据仓库的构建，能够在 Hive 中构建数据仓库；
- 掌握分析招聘信息，能够灵活运用 HiveQL 语句对大数据职位进行分析。

在当今数字化时代，数据已经成为企业和组织的宝贵资产。大数据的兴起为各行各业带来了巨大的机遇和挑战。而数据分析作为从海量数据中提取有价值信息的关键过程，在制定决策、发展业务和获得竞争优势方面发挥着重要作用。本章将详细介绍如何对经过数据预处理的招聘信息进行数据分析。

5.1 数据分析概述

数据分析是对数据进行收集、处理、整理和解释的过程，其目标是从数据中提取有意义的信息，帮助企业和组织更好地理解市场、客户、产品和业务，为正确决策提供支持。数据分析的方法可以分为描述性分析、预测性分析、诊断性分析和推荐性分析，下面将介绍这 4 种数据分析方法的概念。

1. 描述性分析

描述性分析是一种对数据进行汇总、展示和解释的方法，它可以帮助我们更好地理解数据，发现数据的基本特征和分布规律。描述性分析主要依靠统计指标，例如平均值、中位数、众数、累加值等，这些指标反映了数据的集中趋势和离散程度等。

2. 预测性分析

预测性分析是一种基于历史数据和数学模型，对未来可能发生的情况进行预测的方法，它可以利用数据的规律，为决策提供参考，降低不确定性。预测性分析的常用方法有回归分析、时间序列分析、分类分析、聚类分析、关联规则分析等。

3. 诊断性分析

诊断性分析是一种深入分析数据的方法，它可以发现数据中存在的问题，并从中提取

有助于业务决策的见解。诊断性分析常用的方法包括假设检验、方差分析、因子分析、主成分分析、决策树分析等。

4. 推荐性分析

推荐性分析是一种根据数据分析的结果，给出改进建议的方法，它可以利用数据的价值，指导行动的选择，从而提高效率。推荐性分析常用的方法有优化模型、评价模型、决策模型、推荐系统等。

5.2　部署 Hive

1. Hive 的部署模式

在进行数据分析之前，须要先部署 Hive。Hive 支持三种部署模式：内嵌模式、本地模式和远程模式，具体介绍如下。

（1）内嵌模式。

内嵌模式使用 Hive 内置的 Derby 数据库存储元数据。这是 Hive 最简单的部署模式，适用于测试环境。然而，内嵌模式的缺点是元数据不共享，因此当一个客户端连接 Hive 进行操作时，其他客户端无法基于相同的元数据操作 Hive。

（2）本地模式。

本地模式将 Hive 默认的 Derby 数据库替换为关系数据库，如 MySQL，以解决元数据不共享的问题。但本地模式部署的 Hive 具有一定的局限性，因为用户只能在 Hive 部署的服务器上连接 Hive 进行操作，无法在其他服务器上连接 Hive。

（3）远程模式。

远程模式也使用关系数据库存储 Hive 的元数据，但可以解决本地模式部署的局限性。换句话说，用户可以在任意服务器上远程连接 Hive 进行操作。Hive 提供了两种服务供用户实现远程连接，它们分别是 MetaStore 服务和 HiveServer2 服务。

2. Hive 的部署过程

本项目不涉及远程连接 Hive 的相关操作。因此，我们将基于本地模式在虚拟机 Hadoop3 上部署 Hive，具体操作步骤如下。

（1）安装 MySQL。

本项目采用在线安装 MySQL 的方式，在虚拟机 Hadoop3 中安装 MySQL 8.0 版本，具体操作步骤如下。

安装用于从网络上下载文件的工具 wget，以便获取 MySQL 8.0 版本的源配置文件。在虚拟机上执行如下命令。

```
$yum -y install wget
```

使用 wget 工具下载 MySQL 8.0 版本的源配置文件 mysql80-community-release-el9-1.noarch.rpm，该文件是一个 RPM 包。在虚拟机的/export/software 目录中执行如下命令。

```
$wget http://dev.mysql.com/get/mysql80-community-release-el9-1.noarch.rpm
```

上述命令执行完成后,会自动将 MySQL 8.0 版本的源配置文件下载到虚拟机的/export/software 目录中,如图 5-1 所示。

图 5-1 下载 MySQL 8.0 版本的源配置文件

在图 5-1 中,出现"已保存'mysql80-community-release-el9-1.noarch.rpm'"的提示信息,说明 MySQL 8.0 版本的源配置文件下载成功。

使用 yum 工具安装 MySQL 8.0 版本的源配置文件,将 MySQL 8.0 版本的软件包和存储库添加到系统中。在虚拟机的/export/software 目录中执行如下命令。

```
$yum -y install mysql80-community-release-el9-1.noarch.rpm
```

上述命令执行完成后,会自动安装 MySQL 8.0 版本的源配置文件,如图 5-2 所示。

图 5-2 安装 MySQL 8.0 版本的源配置文件

在图 5-2 中,出现"已安装:mysql80-community-release-el9-1.noarch"的提示信息,说明 MySQL 8.0 版本的源配置文件已安装成功。

使用 yum 工具安装 MySQL。在虚拟机上执行如下命令。

```
$ yum -y install mysql-community-server
```

上述命令执行完成后，会自动安装 MySQL，如图 5-3 所示。

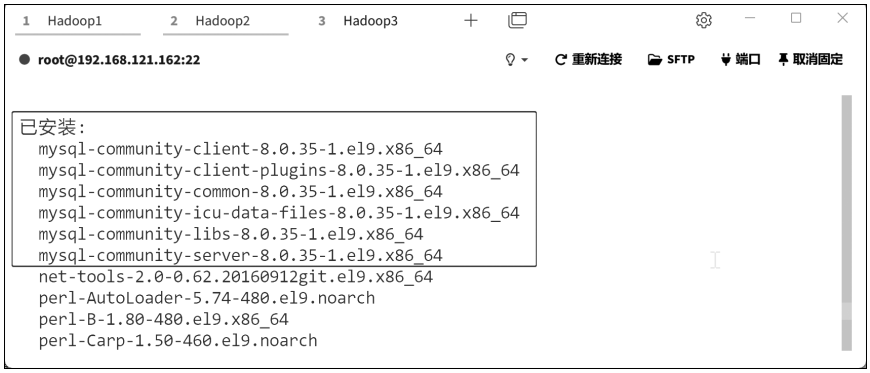

图 5-3　安装 MySQL

在图 5-3 中，出现“已安装：mysql-community……”的提示信息，说明 MySQL 已安装成功。如果出现 GPG 检验失败的错误信息，可以先执行 yum update 命令，更新系统中已安装的软件包，然后再尝试安装 MySQL。

（2）启动 MySQL 服务。

在使用 MySQL 之前，必须启动 MySQL 服务。在虚拟机上执行如下命令。

```
$ systemctl start mysqld
```

（3）查看 MySQL 服务运行状态。

通过查看 MySQL 服务的运行状态，以确认 MySQL 是否可用。在虚拟机上执行如下命令。

```
$ systemctl status mysqld
```

上述命令执行完成的效果如图 5-4 所示。

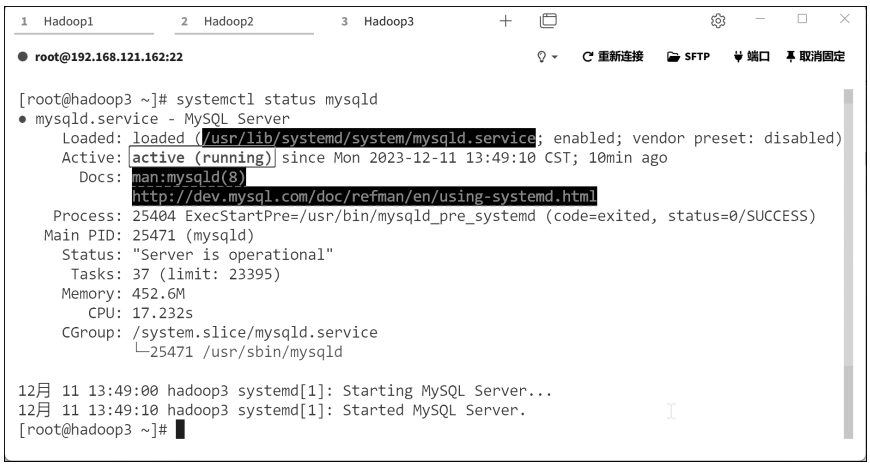

图 5-4　查看 MySQL 服务的运行状态

在图5-4中,出现active(running)的提示信息,说明MySQL服务处于启动状态。

(4) 查看初始密码。

MySQL安装完成后,默认会为本地用户root提供一个初始密码,以便登录MySQL进行相关配置。在虚拟机上执行如下命令查看初始密码。

```
$grep 'temporary password' /var/log/mysqld.log
```

上述命令执行完成的效果如图5-5所示。

图5-5 查看初始密码

从图5-5中可知,本地用户root的初始密码为b&5f9G>wZZio。需要说明的是,每次安装MySQL时,本地用户root的初始密码都会有所不同。

(5) 登录MySQL。

通过本地用户root登录MySQL,对其进行必要的配置。在虚拟机上执行如下命令。

```
$mysql -uroot -p
```

上述命令执行完成的效果如图5-6所示。

图5-6 登录MySQL(1)

在图5-6中的"Enter password:"位置输入本地用户root的初始密码,然后按下Enter键,如图5-7所示。

在图5-7中,出现mysql>提示符说明成功登录了MySQL,并进入MySQL命令行界面,在该界面中可以执行SQL语句来操作MySQL。

(6) 修改密码。

在对MySQL进行配置之前,需要修改本地用户root的初始密码。这里将本地用户root的初始密码修改为Itcast@2023。在MySQL命令行界面中执行如下命令。

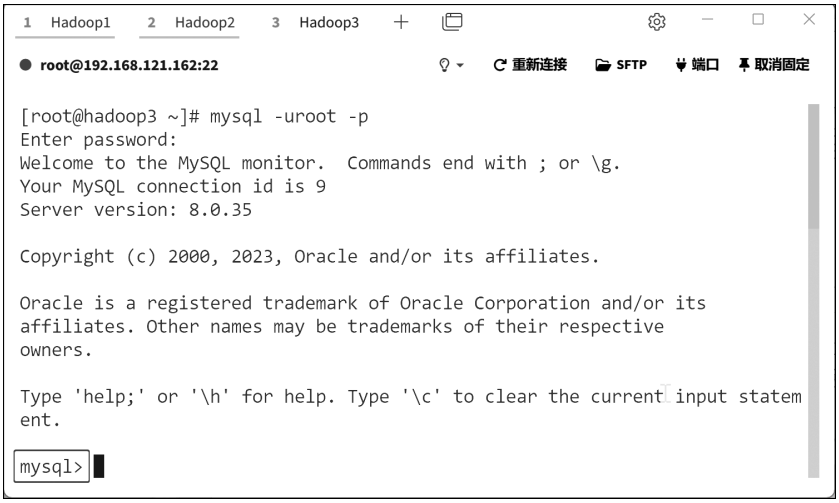

图 5-7　登录 MySQL(2)

```
mysql>ALTER USER 'root'@'localhost' IDENTIFIED BY 'Itcast@2023';
```

上述命令执行完成后,须重新加载 MySQL 的权限表,以确保对用户权限的更改立即生效。在 MySQL 命令行界面中执行如下命令。

```
mysql>FLUSH PRIVILEGES;
```

上述命令执行完成后,可以在 MySQL 命令行界面执行“exit;”命令退出登录。然后,再次使用本地用户 root 登录 MySQL,以验证密码是否成功修改。

(7) 添加用户。

本项目后续实现数据可视化时,需要通过远程登录 MySQL 来读取数据。因此,我们需要在 MySQL 中添加一个名为 itcast 的用户,用于远程登录 MySQL。在 MySQL 命令行界面中执行下列命令。

```
--添加一个名为 itcast 的用户,并指定密码为 Itcast@2023
mysql>CREATE USER 'itcast'@'%' IDENTIFIED BY 'Itcast@2023';
--授予用户 itcast 对所有数据库和表拥有权限,并允许该用户通过任何主机进行远程登录
mysql>GRANT ALL PRIVILEGES ON *.* TO 'itcast'@'%' WITH GRANT OPTION;
```

上述命令执行完成后,需要重新加载 MySQL 的权限表,以确保对用户权限的更改立即生效。

(8) 上传 Hive 安装包。

将 Hive 安装包 apache-hive-3.1.3-bin.tar.gz 上传至虚拟机的/export/software 目录中。

(9) 安装 Hive。

采用解压缩方式将 Hive 安装至/export/servers 目录。在虚拟机上执行如下命令。

```
$tar -zxvf /export/software/apache-hive-3.1.3-bin.tar.gz \
-C /export/servers/
```

上述命令执行完成后,在虚拟机的/export/servers 目录会看到一个名称为 apache-hive-3.1.3-bin 的目录。为了便于后续使用 Hive,这里将 Hive 的安装目录重命名为 hive-3.1.3,在虚拟机的/export/servers 目录中执行如下命令。

```
$mv apache-hive-3.1.3-bin hive-3.1.3
```

(10) 同步依赖文件。

Hive 和 Hadoop 都依赖 Guava 库来实现一些功能。然而,在 Hive 3.1.3 和 Hadoop 3.3.0 中,这二者使用的 Guava 库版本不一致。具体而言,Hive 3.1.3 使用 Guava 库的版本为 19.0,而 Hadoop 3.3.0 使用 Guava 库的版本为 27.0。为了避免启动 Hive 时出现依赖文件冲突的问题,我们需要统一 Hive 和 Hadoop 使用 Guava 库的版本。鉴于向下兼容性的考虑,这里用 Hadoop 3.3.0 所使用的 Guava 库替换 Hive 3.1.3 中的 Guava 库,具体操作步骤如下。

首先,进入 Hadoop 存放 Guava 库的目录。在虚拟机上执行如下命令。

```
$cd /export/servers/hadoop-3.3.0/share/hadoop/common/lib
```

然后,将 Hadoop 使用的 Guava 库 guava-27.0-jre.jar 复制到 Hive 存放 Guava 库的目录。在虚拟机上执行如下命令。

```
$cp guava-27.0-jre.jar /export/servers/hive-3.1.3/lib
```

最后,删除 Hive 自带的 Guava 库 guava-19.0.jar。在虚拟机上执行如下命令。

```
$rm -fr /export/servers/hive-3.1.3/lib/guava-19.0.jar
```

(11) 添加 MySQL 驱动包。

在使用 MySQL 存储 Hive 的元数据时,Hive 需要通过 JDBC 与 MySQL 建立连接。因此,我们需要将 MySQL 8.0 版本的 JDBC 驱动包 mysql-connector-j-8.0.31.jar 上传到 Hive 安装目录的 lib 目录中。

(12) 创建 Hive 的配置文件。

Hive 默认不提供用户可编辑的配置文件,而是提供一个名为 hive-default.xml.template 的模板文件供用户参考。为了创建 Hive 的配置文件,我们可以在 Hive 存放配置文件的目录中创建一个名为 hive-site.xml 的文件。在虚拟机的/export/servers/hive-3.1.3/conf 目录中执行如下命令。

```
$touch hive-site.xml
```

(13) 编辑 Hive 的配置文件。

使用 vi 编辑器来编辑 Hive 的配置文件 hive-site.xml，在该文件中添加 Hive 的配置信息。配置文件 hive-site.xml 修改完成后的效果如文件 5-1 所示。

文件 5-1　hive-site.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <property>
        <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExist=true
</value>
    </property>
    <property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>com.mysql.cj.jdbc.Driver</value>
    </property>
    <property>
        <name>javax.jdo.option.ConnectionUserName</name>
        <value>root</value>
    </property>
    <property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>Itcast@2023</value>
    </property>
</configuration>
```

在文件 5-1 中，参数 javax.jdo.option.ConnectionURL 用于配置 JDBC 连接 MySQL 的地址。参数 javax.jdo.option.ConnectionDriverName 用于配置驱动程序的名称。参数 javax.jdo.option.ConnectionUserName 和 javax.jdo.option.ConnectionPassword 分别用于配置连接 MySQL 的用户和密码。

在配置文件 hive-site.xml 中添加 Hive 的配置信息后，保存并退出编辑。

（14）配置 Hive 的系统环境变量。

为了方便后续使用 Hive，我们需要在虚拟机中配置 Hive 的系统环境变量。在虚拟机中，使用 vi 编辑器来编辑系统环境变量文件 profile，在文件的末尾添加以下内容。

```
export HIVE_HOME=/export/servers/hive-3.1.3
export PATH=$PATH:$HIVE_HOME/bin
```

在系统环境变量文件 profile 中添加上述内容后，保存并退出编辑。然后，初始化虚拟机的系统环境变量，使系统环境变量文件 profile 中修改的内容生效。

（15）初始化 Hive 的元数据。

在 MySQL 中初始化 Hive 的元数据，创建 Hive 所需的数据库和表结构。在虚拟机

上执行如下命令。

```
$schematool -initSchema -dbType mysql
```

上述命令执行完成的效果如图5-8所示。

图5-8 初始化Hive的元数据

在图5-8中,出现Initialization script completed和schemaTool completed的提示信息,说明在MySQL中成功初始化了Hive的元数据。

(16)连接Hive。

通过Hive自带的命令行工具CLI连接Hive。在虚拟机上执行如下命令。

```
$hive
```

上述命令执行完成的效果如图5-9所示。

图5-9 连接Hive

在图5-9中,出现hive>提示符说明成功连接Hive,并进入CLI命令行界面,在该界面可以执行HiveQL语句来操作Hive。

5.3 数据仓库

5.3.1 数据仓库简介

数据仓库是一个面向主题、相对稳定和反映历史变化、集成的数据集合,用于支持企业或组织的决策分析。对数据仓库的定义指出了其具有以下4个特点。

1. 数据仓库是面向主题的

数据库管理系统以业务流程为导向组织数据，主要面向事务处理。然而，数据仓库以需求分析为导向组织数据，主要面向数据分析。数据仓库中的数据按照主题进行组织，主题是指用户在使用数据仓库进行决策时关注的重点方面。举例来说，可以将公司的销售数据划分为一个主题，从而实现与销售相关的数据分析。

2. 数据仓库是相对稳定的

数据仓库中的数据反映一段时间内的历史数据。数据进入数据仓库后通常会被保留较长时间。由于数据仓库的目标是支持决策分析而非事务处理，所以数据修改和删除操作相对较少，使得不同数据源集成到数据仓库的过程相对稳定。

3. 数据仓库是反映历史变化的

数据仓库记录的数据是历史数据，这些数据反映了企业从过去某个时间点到目前各阶段的信息，方便企业对业务进行分析，从而改善业务经营决策以及预测企业未来趋势。

4. 数据仓库是集成的

数据仓库的数据通常来自不同的数据源，需要从这些数据源中抽取与需求分析相关的数据。为了确保数据的一致性，需要对来自不同数据源的数据进行转换处理，最终将其加载到数据仓库中，实现数据的集成。

5.3.2　数据仓库设计

1. 数据仓库的分层结构

数据仓库分层是数据仓库设计的基础和依据，它是对数据仓库内的数据进行有效组织和管理的方法。通常情况下，数据仓库内的数据分为 3 层，即源数据层（operational data store，ODS）、数据仓库层（data warehouse，DW）和数据应用层（application data service，ADS），具体介绍如下。

（1）源数据层。

源数据层存储的数据是数据仓库的基础数据，该层存储的数据抽取自不同的数据源，抽取的这些数据通常会进行诸如去噪、去重、标准化等一系列转换操作后才会加载到源数据层。不过在某些应用场景中，为了确保源数据层存储数据的原始性，也可以直接将从不同数据源抽取的数据加载到源数据层，不进行任何转换操作。

（2）数据仓库层。

数据仓库层存储的数据是对源数据层中数据的轻度汇总，所谓轻度汇总就是按照一定的主题去组合这些数据。数据仓库层从上到下，又可以细分为明细层（data warehouse detail，DWD）、中间层（data warehouse mart，DWM）和业务层（data warehouse service，DWS），具体介绍如下。

① 明细层的作用是根据业务需求对源数据层的数据进行进一步转换，不过该层的数据粒度与源数据层的数据粒度保持一致。

② 中间层的作用是在明细层的基础上，对数据做一些轻微的聚合操作，生成一系列的中间表，从而提高公共指标的复用性，减少重复工作。

③ 业务层的作用是在明细层和中间层的基础上，对某个主题的数据进行汇总，主要