# Python+MySQL数据库操作

本文介绍如何利用python来对MySQL数据库进行操作，本文将主要从以下几个方面展开介绍：

1. 数据库介绍
2. MySQL数据库安装和设置
3. Python操作MySQL

* 在Python3.X上安装MySQL驱动
* 创建数据库连接
* 创建数据表
* 增、改、删、查
* 分组、聚合
* 按批量读取和处理数据

1. 小结

## 1 数据库介绍

数据库（Database）是按照数据结构来组织、存储和管理数据的仓库，能直接通过条件快速查询到指定的数据。随着信息技术和市场的发展，数据管理不再仅仅是存储和管理数据，而转变成用户所需要的各种数据管理的方式。

目前，数据库主要有两种形式，一种是非关系型数据库，另一种是关系型数据库。目前，我们用得非常广泛的一种数据库类型是关系型数据库，它可以分为以下几种：

* Oracle:付费产品，主要是银行在用（万一出错了有Oracle背锅）
* DB2:付费产品，IBM产品
* SQL Sever：付费产品，微软产品，windows专用
* PostgreSQL：免费产品，主要是高校学术上使用
* MySQL：大众，免费，开源

作为手无寸金的大学生，我们应该用哪种数据库呢？当然是MySQL。一方面是因为MySQL免费，另一方面是因为普及率最高，出了错，可以很容易找到解决方法。而且，围绕MySQL有一大堆监控和运维的工具，安装和使用很方便。所以，本文接下来也会介绍如何用Python来操作MySQL。

## 2 MySQL安装和设置

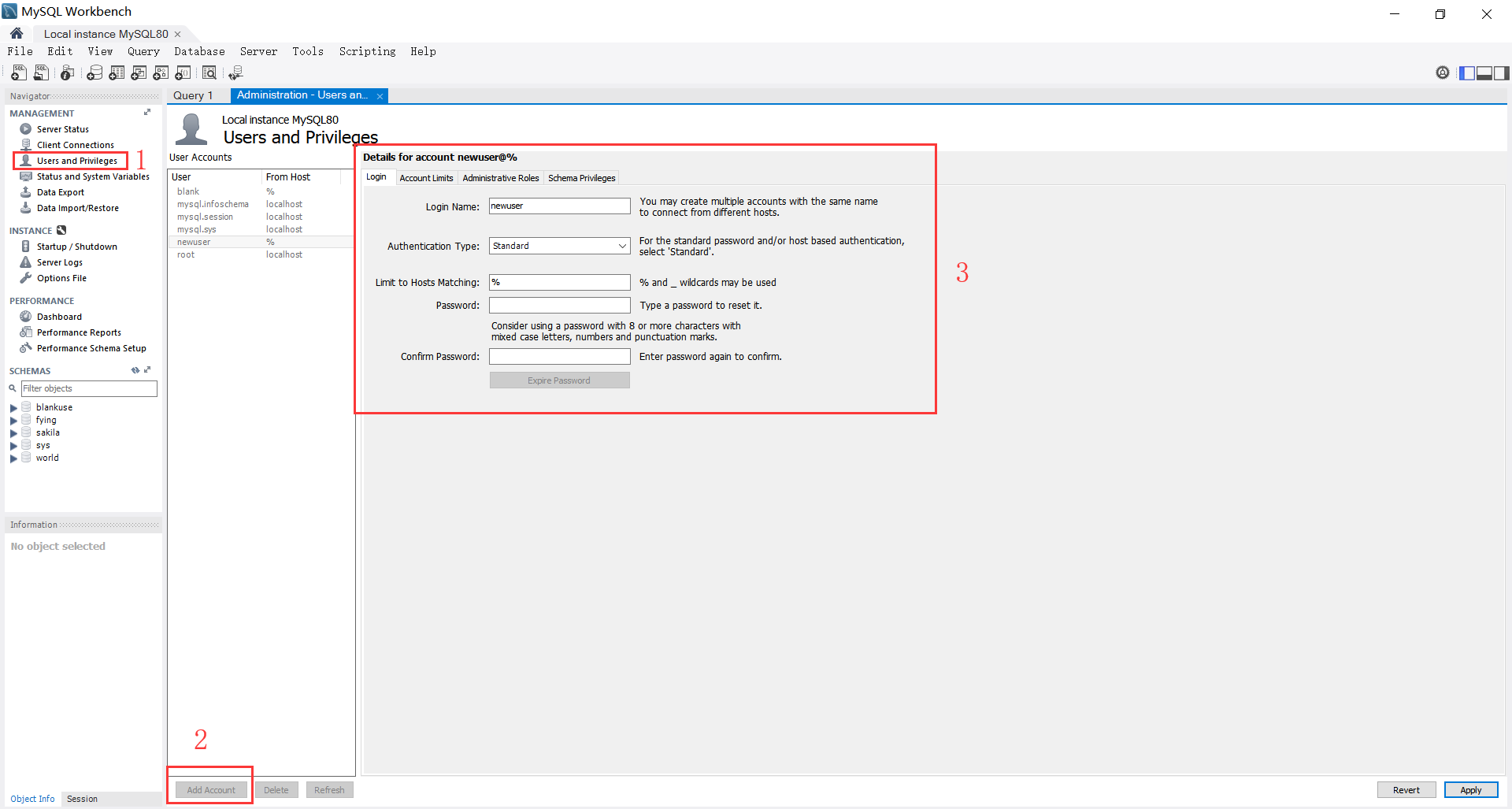
### 2.1 下载安装

**STEP1**：从MySQL官方网站上下载最新的MySQL Installer 8.0.14版本，下载链接为：https://dev.mysql.com/downloads/installer/

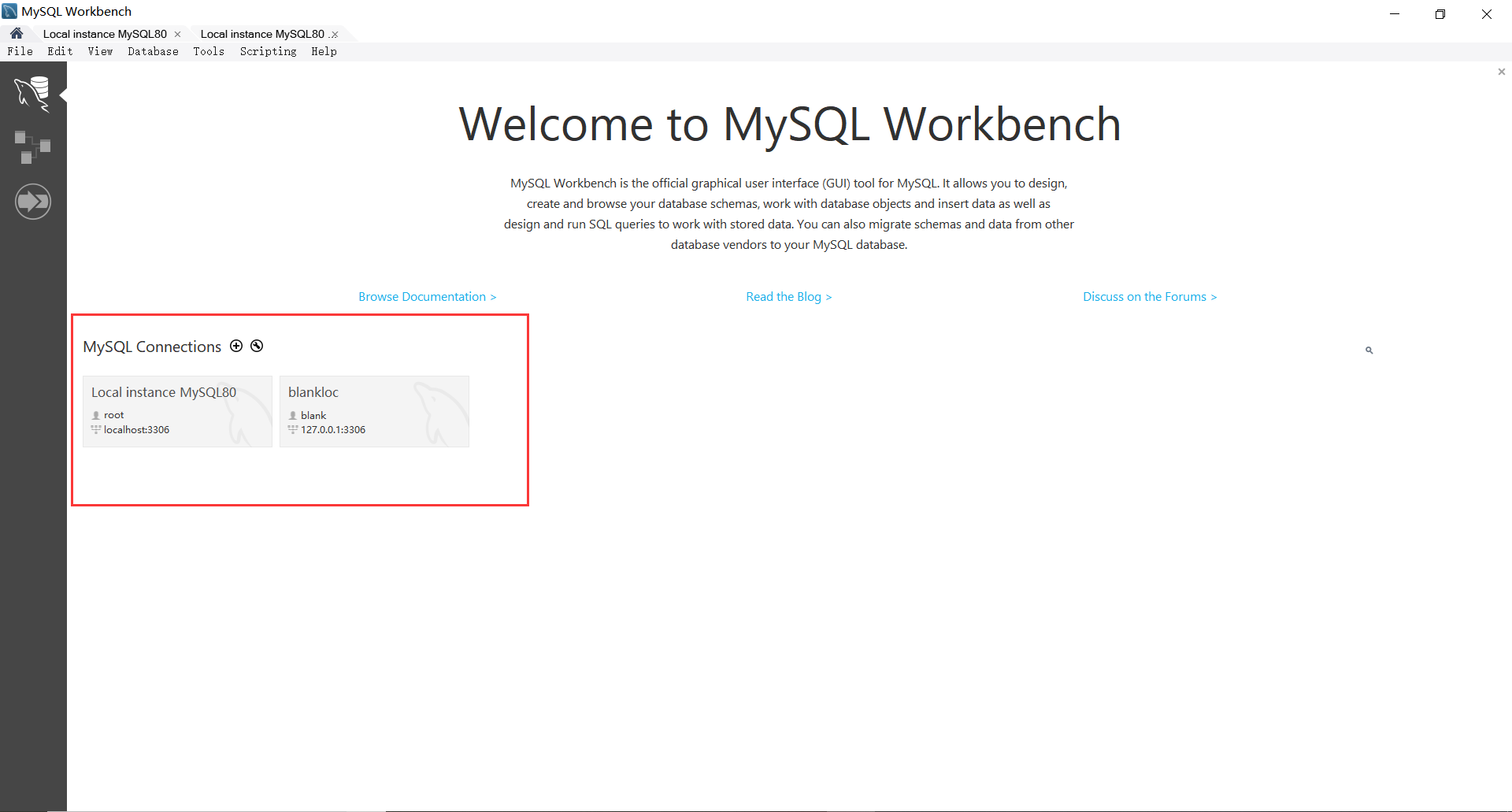
**STEP2**：按照指示操作默认安装，在安装时，MySQL会要求我们设置一个本地登陆账号，账号名一般命为root，端口为3306，自定义一个password即可。

### 2.2 MySQL workbench创建用户与授权：

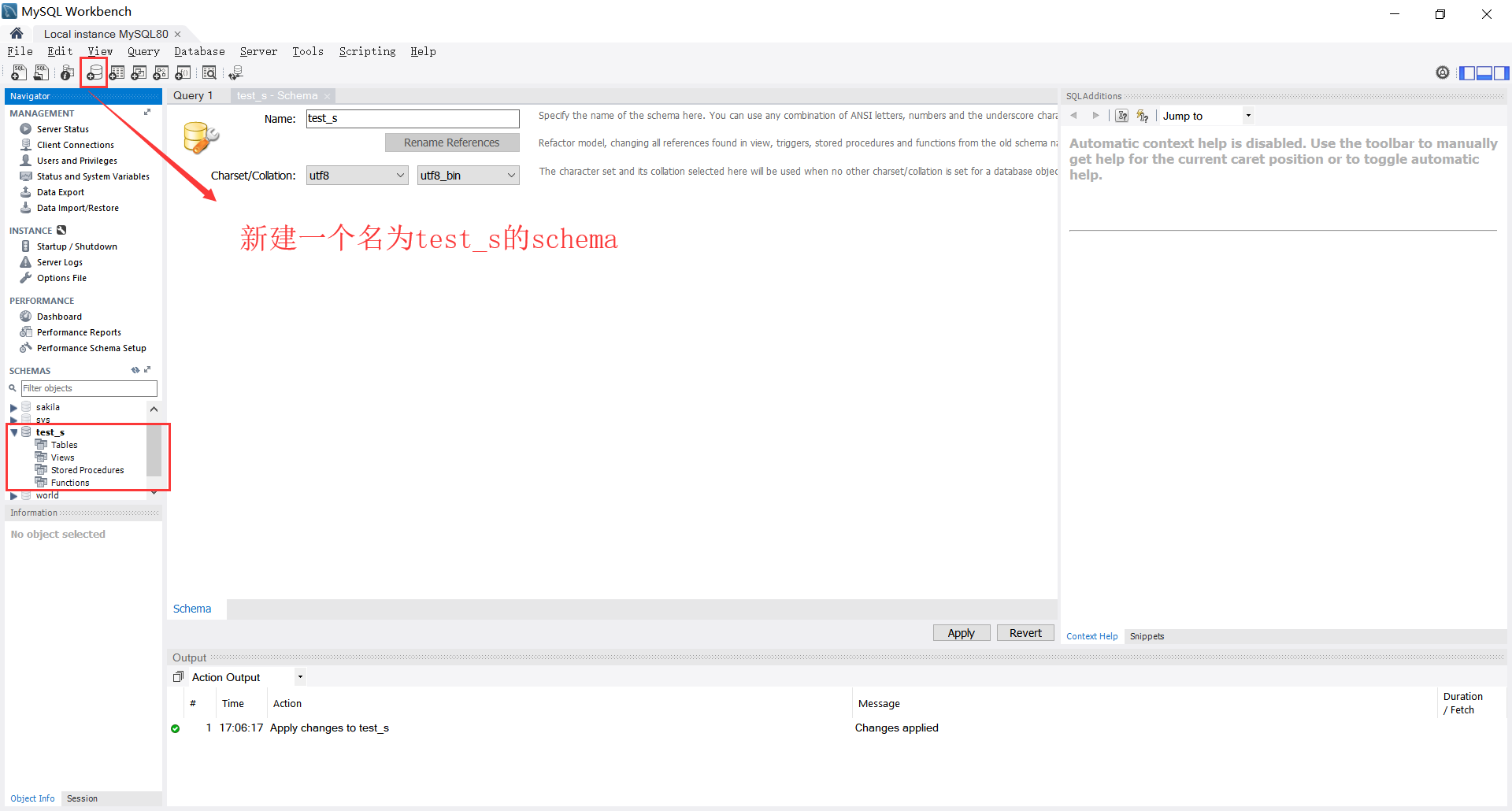
**STEP1**：创建新用户。以root用户登录MySQL workbench，先选择users and privileges，然后选择下方的add account去添加用户。



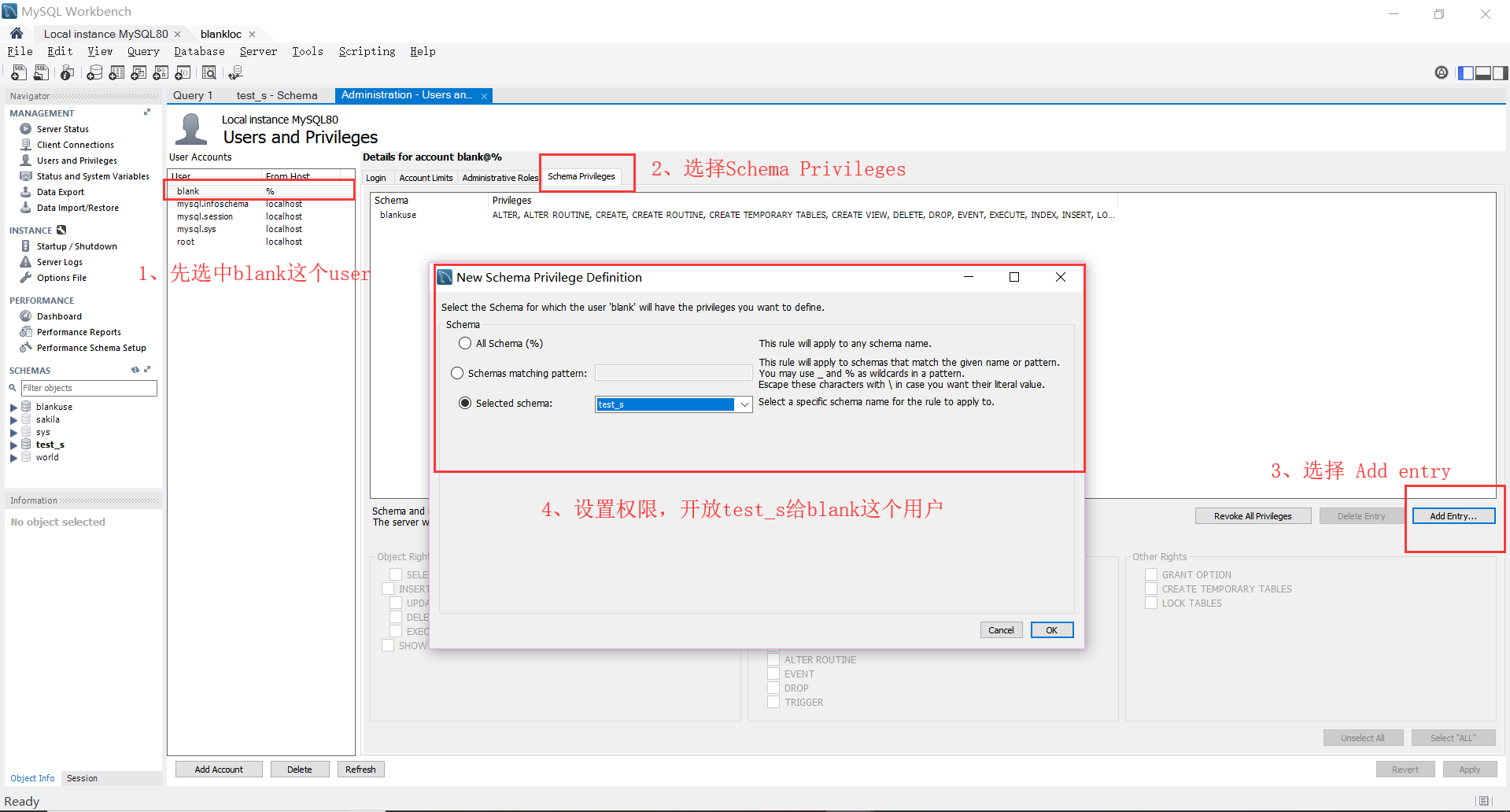
**STEP2**：我们以创建好的blank为例，创建好后在左边的表中出现了blank这个用户。返回MySQL Workbench主页，新建MySQL Connection，用户名和密码为新创建的用户名和密码，这个时候，我们就能看到除了root用户外，还能看到新创建的blank这个user账户了。



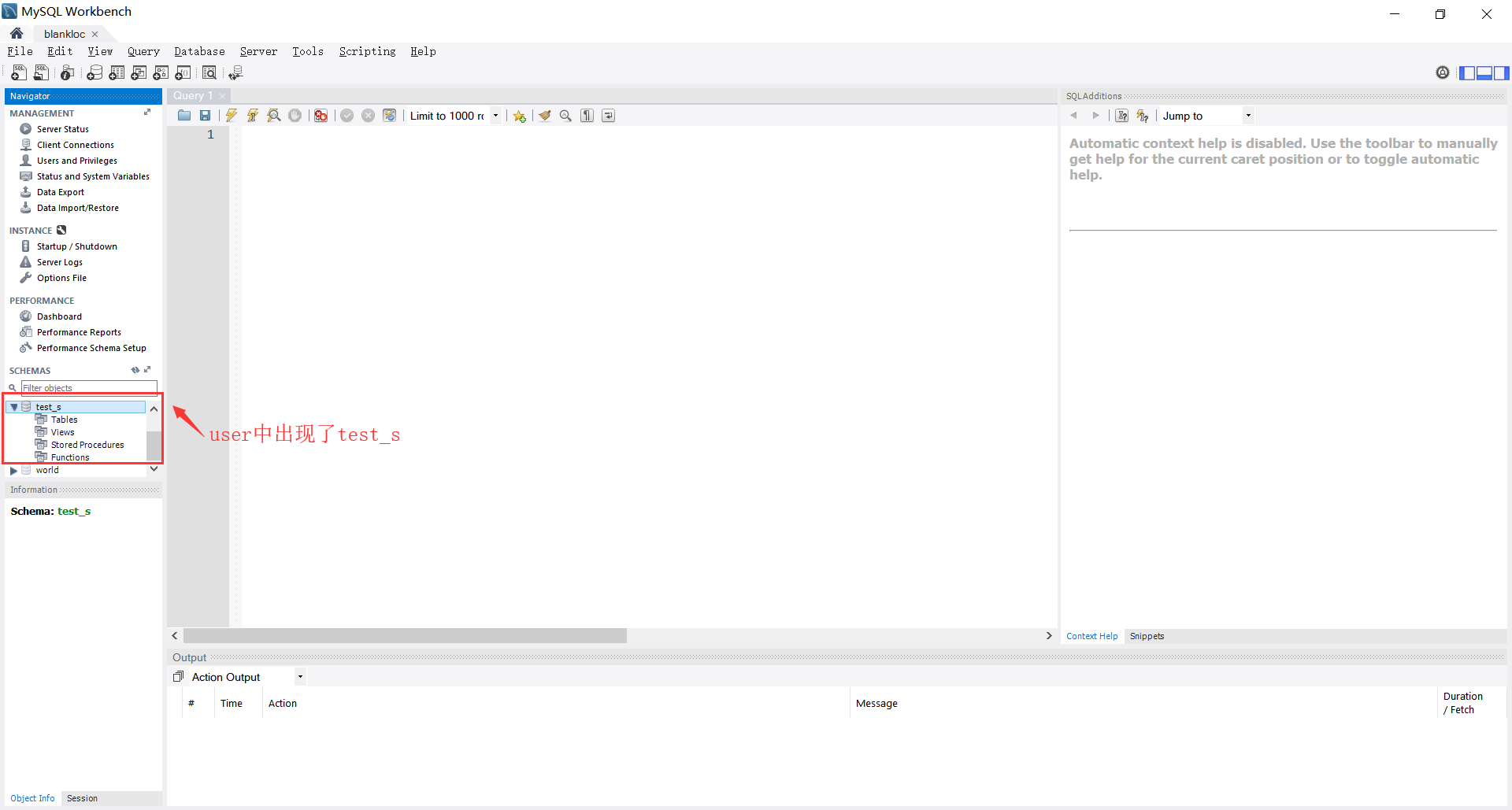
**STEP3：**创建数据库和数据表。在root账户中新建一个名字为test\_s的Schema，然后我们会在左下角的schemas中看到我们新建的schema。对于Schema，在MySQL中，它等同于database，它是数据库对象的集合，这个集合包括了各种对象，如Tables（表）、Views（视图）、Sorted Procedures（存储过程）、Functions等，我们可以选中Talbe，点击鼠标右键，选择creat table，即可在该数据库下创建数据表。创建过程和创建数据库类似。



**STEP4：**设置用户权限：而当我们用blank连接数据库时，是没有test\_s这个schema的，这个时候我们需要通过root开放权限给blank这个用户。返回root用户操作选项卡，选择users and privileges，选中blank用户，再选择schema privileges，点击add entry；在弹出来的窗口中选择权限范围，这里我们选择指定的test\_s给blank这个user，点击ok；双击tets\_s，privileges全部授予,点击select all。



**STEP5：**进入主界面，重新进入blank用户操作选项卡，我们会在schemas中看到test\_s这个schema。blank这个用户可以对root授权的test\_s这个schema中的表进行操作。



至此，我们完成了MySQL中用户的新建和授权。

## 3 Python操作MySQL

目前，关于Python操作数据库主要有以下几种方法：

* **MySQLdb的使用**

MySQLdb是用于Python连接MySQL数据库的接口，它实现了Python数据库API规范V2.0，基于MySQL C API上建立的，目前只支持Python2.x。

* **PyMySQL的使用**

PyMySQL是Python中用于连接MySQL服务器的一个库，它支持Python3.x，是一个纯Python写的MySQL客户端，它的目标是替代MySQLdb。PyMySQL在MIT许可下发布。

* **mysql.connector 的使用**

由于 MySQL 服务器以独立的进程运行，并通过网络对外服务，所以，需要支持 Python 的 MySQL 驱动来连接到 MySQL 服务器。

目前，有两个 MySQL 驱动：

1. mysql-connector-python：是 MySQL 官方的纯 Python 驱动
2. MySQL-python ：是封装了 MySQL C驱动的 Python 驱动

* **SQLAlchemy的使用**

是一种ORM（Object-Relational Mapping）框架，将关系数据库的表结构映射到对象上，隐藏了数据库操作背后的细节，简化了数据操作。

### 3.1 在Python3.X上安装MySQL驱动

**STEP1**：由于MySQL官方提供了mysql-connector-python驱动。安装时，在Anaconda Prompt中输入：

**STEP2**：使用以下代码来测试mysql-connector是否安装成功：



如果没有产生错误，则表明安装成功。

### 3.2 创建数据库连接

这里连接的是我之前创建的blank这个user。如果数据库已经存在的话，我们可以直接连接；如果数据库不存在，直接连接则会报错，这个时候我们就需要创建一个数据库，创建数据库可以在MySQL Workbench中创建，也可以在python中使用"CREATE DATABASE"语句，在本实验中，我们使用已经在MySQL workbench中已经建好的test\_s这个数据库。



### 3.2 创建数据表

**STEP1:**当Python 和数据之间的连接建立起来之后，要操作数据库，就需要让 Python对数据库执行SQL语句。创建数据表我们使用"CREATE TABLE"语句，在test\_s这个数据库中创建一个叫做customers的表格，其中包含id、name、address、sex、age、sl这六个columns。Python是通过游标执行SQL语句的，所以，连接建立之后，就要利用连接对象得到游标对象。

cursor()：表示游标

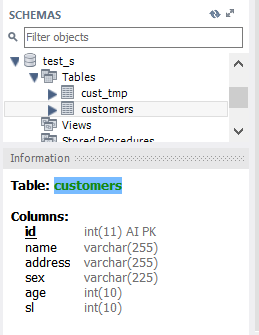
execute()：是执行语句

**STEP2：**一般在创建新表的时候，我们还会设置一个主键（PRIMARY KEY）来方便进行查询工作。创建主键，我们可以用"INT AUTO\_INCREMENT PRIMARY KEY"



VARCHAR()表示的是数据类型，定义的是变长字符串；INT（）表示整型

**STEP3**：执行语句。执行完后，我们可以回到MySQL workbench，可以看到在test\_s下面的customers这个表格，其中Columns为我们创建的id，name，address，sex，age和sl。



**STEP4：**但是，当我们再次执行语句的时候，由于已经创建了"customers"这个表，所以再次执行会报错，这个时候就需要加一个判断，判断这个表是否已经存在于test\_s这个数据库中。



**STEP5：**我们可以用"SHOW TABLES"语句来查看数据表是否已经存在，如果存在就print"table already exists",如果不存在，就print"table does not exist"。



**STEP6**：上面的语句只是为了帮助我们判断是否有同名表，当我们要新建一个表时，我们可以在这个判断的基础上，在创建新表前删掉数据库内的同名表，再建新表。删除我们用的是"DROP TABLE"，新建表是"CERAYE TABLE"



### 3.3 增、改、删、查

#### 3.3.1 增

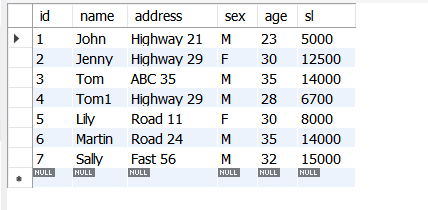
在cutomers表中插入数据用的是"INSERT INTO"语句。

除了用一条条用execute( )插入之外，我们还可以用executemany（）的方式批量插入，也就是val中包含的是一个元组列表，包含我们想要插入的数据。

需要注意的事是：如果数据表格有更新，那么必须用到commit()语句，否则在workbench是看不到插入的数据的。



执行以上代码后，回到workbench，，我们可以看到最终的结果为：

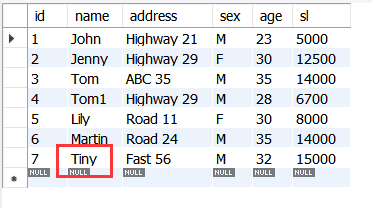


#### 3.3.2 改

在cutomers表中更改数据用的是"UPDATE"语句。例如，我们将最后一条 “Sally”的名字改成“Tiny”：



执行代码，回到workbench我们可以看到结果为：

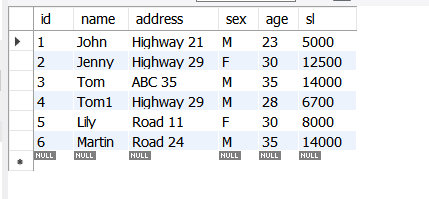


#### 3.3.3 删

关于删，我们在上文提到了删除表格，用的是“DROP TABLE ”语句，“IF EXISTS”关键字是用于判断表是否存在，只有在存在的情况才删除当我们要删除一条数据记录时候，用到的语句是“DELETE FROM”语句。例如：我们想在customers这个表格当中，删除name为Tiny的这一条记录：



执行代码，回到workbench我们可以看到结果为：



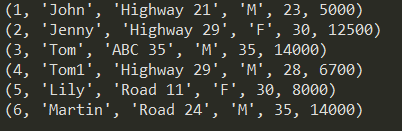
#### 3.3.4 查

* **普通查找**

普通查询数据用的是SELECT语句。例如：我们想查询customers的所有信息，并且进行打印输出：



得到最终结果为：



值得注意的是：fetchall（）表示的是获得所有记录；fetchone（）表示只获取一条数据；fetchmany（size=3）表示获取三条记录；

* **限定条件查找**

为了获取指定条件下的查找结果，我们可以使用where语句。例如：我们想在查询customers的所有信息基础上，输出年龄大于30岁的消费者的信息：



最终得到的结果为：



* **通配符查找**

有时候为了进行模糊查询，可以匹配通配符，通过“LIKE”来进行查找：

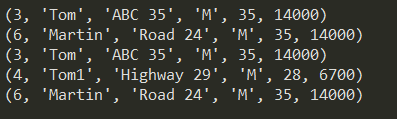
百分号 （%）:代表零个、一个或多个数字或字符；

下划线 （\_）:代表一个单一的数字或字符。

例如：查出所有名字中含有t的记录：



执行代码，我们得到的结果如下：



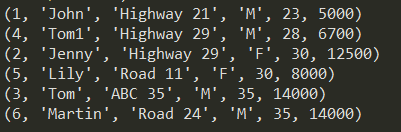
值得注意的是：但是使用Like查询时，即使我们在代码输入的是“t”，执行过程中也会将含有“T”的记录同样输出，即用LIKE匹配通配符对大小写不敏感。为了区分大小写，可以用“GLOB”进行查询。

* **排序**

查询结果排序可以使用 ORDER BY 语句，默认的排序方式为升序，如果要设置降序排序，可以设置关键字 DESC。例如：我们要按照年龄对customers进行升序排列：



执行代码，得到的结果为：

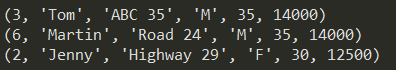


* **LIMIT**

当数据库数量非常大的时候，为了限制查询的数据量，可以采用"LIMIT"语句来指定，比如我们希望在customers表中找出工资最高的三个人：



执行代码，得到结果为：

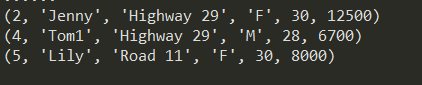


* **二次筛选**

有时候我们在进行一次筛选后，还需要设定一个筛选条件进行二次筛选，我们就可以采用“HAVING”语句。例如：我们希望统计在年龄处于20-30（不包括20岁，但是包括30岁）的人当中，选择薪资大于5000的消费者：



执行代码后，得到的结果如下：



### 3.4 分组聚合

在数据库中，分组常用的语句为“GROUP BY”语句，聚合函数，通常是配合分组进行使用，在数据库中常用的聚合函数为：

1. **COUNT（\*）**：表示计算总行数，括号可以写\*和字段名字
2. **MAX（column）**：表示求此列的最大值
3. **MIN（column）**：表示求此列的最小值
4. **SUM（column）**：表示求此列的和
5. **AVG（column）**：表示求此列的平均值

* **从customers表中统计出男女薪资总和**

以sex为类别进行GROUP BY 分组，加上WHERE来做条件判断。



最终结果为：



* **从customers表中，按性别进行分组，统计出年龄在20-30的消费者的薪资，并且按照薪资高低进行排序**





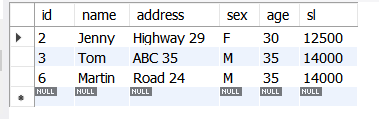
值得注意的是：本例是以sex为类别进行GROUP BY 分组，加上WHERE来做条件判断，加上ORDER BY 排序，但是GROUP BY 的位置必须要在WHERE 之后，在ORDER BY 之前。

### 3.5 分批量读取和处理数据

程序运行的时候，数据都是在内存中的，但是有时候如果数据量太大，内存会装不下，这个时候我们就需要分批从数据库去读取数据，然后再处理，等到处理完了之后，再去读取。比如：我们要从customers当中分批读取和处理薪资大于8000的消费者，并将其存入另一张表中。我们的做法是先新建一个表，然后从数据库当中读取3个，并且将读取的这3个进行处理，处理完读取的这三个后，再去数据库重新读取三个，直到数据库的数据读完为止。

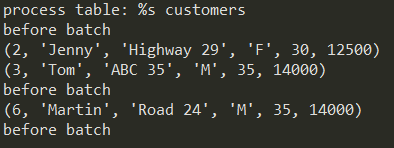


我们回到workbench找到这个新建的表格cust\_tmp，我们可以发现薪资大于8000的消费者都被记录上了：



执行代码，我们可以看到处理的过程如下：

在第一批读取的三条记录中，只有两条是满足薪资大于8000的要求，第二批读取的三条记录中，只有一条满足薪资大于8000的要求，而在第三批读取的三条记录中，没有任何记录是满足薪资大于8000的要求，当没有记录可以读的时候，程序即停止。



值得注意的是：就分批读取的batchsize而言，当batchsize太大时，会导致内存装不下，batchsize太小，会导致每次通过网络连接数据库会很慢。因此，我们选取batchsize大小的原则是在内存够用的前提下尽可能的大，在真实的业务场景下，建议每次读取100以上，当内存够用的话，也可以增加至几千上万条。

## 4 小结

本文介绍了Python+MySQL的基本操作，包括如何安装Mysql，如何装驱动，如何创建连接以及对数据库进行增删改查、分组聚合以及批量读取和处理等操作。但是，本文涉及到的只是对单表进行操作，只是数据库操作的冰山一角；在实际的开发和工作环境中，需要根据实际内容对多表进行操作，这部分请持续关注数据魔术师关于数据库的后期推文。