Databases and Queries in MySQL

Ankur Malik, Haotian Xia, Haoming Deng, Jennifer Park, Russell Liu

Abstract:

In this methods vignette, we explore how to configure a database and write queries in R. First, we go over the required software and packages. Then, we include steps to create a new database, make configurations, and enter new data into an existing database using some sample data. Lastly, we explain how to write queries by introducing some common functions, tables, syntax and the filtering clause in SQL.

Introduction:

This is a tutorial on how to configure a database in R.

Goal: Make people feel comfortable in basic programming with a database and instruct them on how to extract the data they want.

Target: People who do not have database programming experience.

Scope: SQL and R.

Software: R version 4.2.0+, RStudio version 2022.07.1+, MySQL version 8.0.31+, MySQL Workbench version 8.0.31 (Optional)

CRAN Packages: RMariaDB, dbplyr, dplyr

RMariaDB is a DBI-compliant database interface and MariaDB driver for R, meaning that it uses SQL statements wrapped in R functions in order to access MySQL databases. SQL is one of the most popular database languages for people to extract data information. However, it has its own syntax, which can hard to pick up quickly. Since coding backgrounds may vary, we also introduce dbplyr, a CRAN package that translates the most common R functions to their SQL equivalents. Thus, users can access remote database tables as if they are in-memory R data frames without having to write raw SQL code themselves.

However, dbplyr does not cover all use cases and is not as powerful as SQL. Therefore, this project will primarily focus on SQL instruction and demonstrate how dbplyr works in some cases.

By the end of this vignette, you will know how to apply database methods successfully in real projects.

What is a Query, SQL, and MySQL?

A **query** is a statement which enables us to extract data or information from a particular database table. We are writing a statement or command which allows us to extract a specific piece of information from our database. Now you may wonder what a database is? A **database** is the collection of structured data stored within a computer system. We use database management systems (*DBMS*) to control a database.

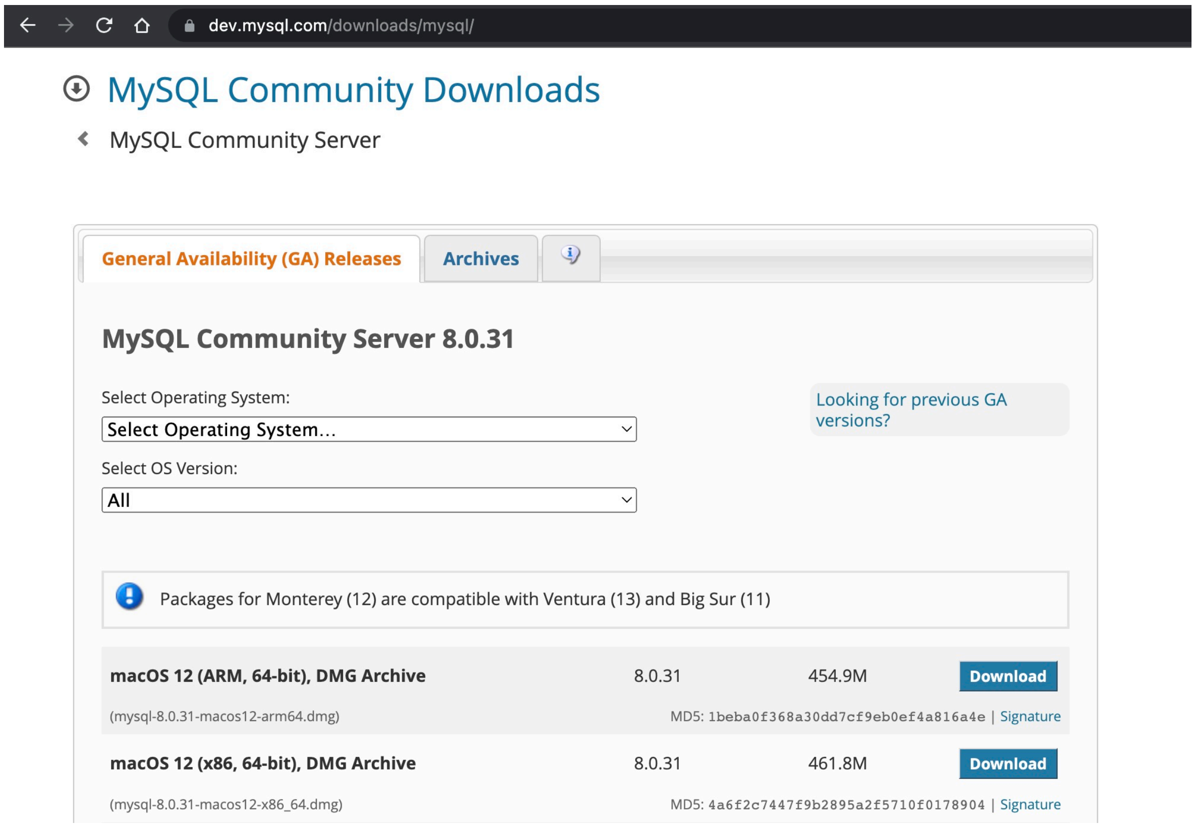
SQL is an acronym for *standard query language*. SQL is considered to be a programming language which can be used to manipulate, query, access, and control a database or table.

In this project, we are focusing on a specific kind of SQL called MySQL**.** **MySQL** is defined a relational database management system based on SQL. It differs from other versions of SQL primarily based on syntax, functionality, and purpose.

Assuming R and RStudio have already been installed, let's begin with how to install the MySQL Server.

Installing the MySQL Server

1. Download MySQL Community Server 8.0.31. Be sure to select the correct operating system and processor for your computer. For Macbook M1/M2, it is "macOS 12 (ARM, 64-bit), DMG Archive". You can start the download without creating an Oracle Web Account.


dev.mysql.com/downloads/mysql/

MySQL Community Downloads

MySQL Community Server

General Availability (GA) Releases | Archives

MySQL Community Server 8.0.31

Select Operating System: Select Operating System...

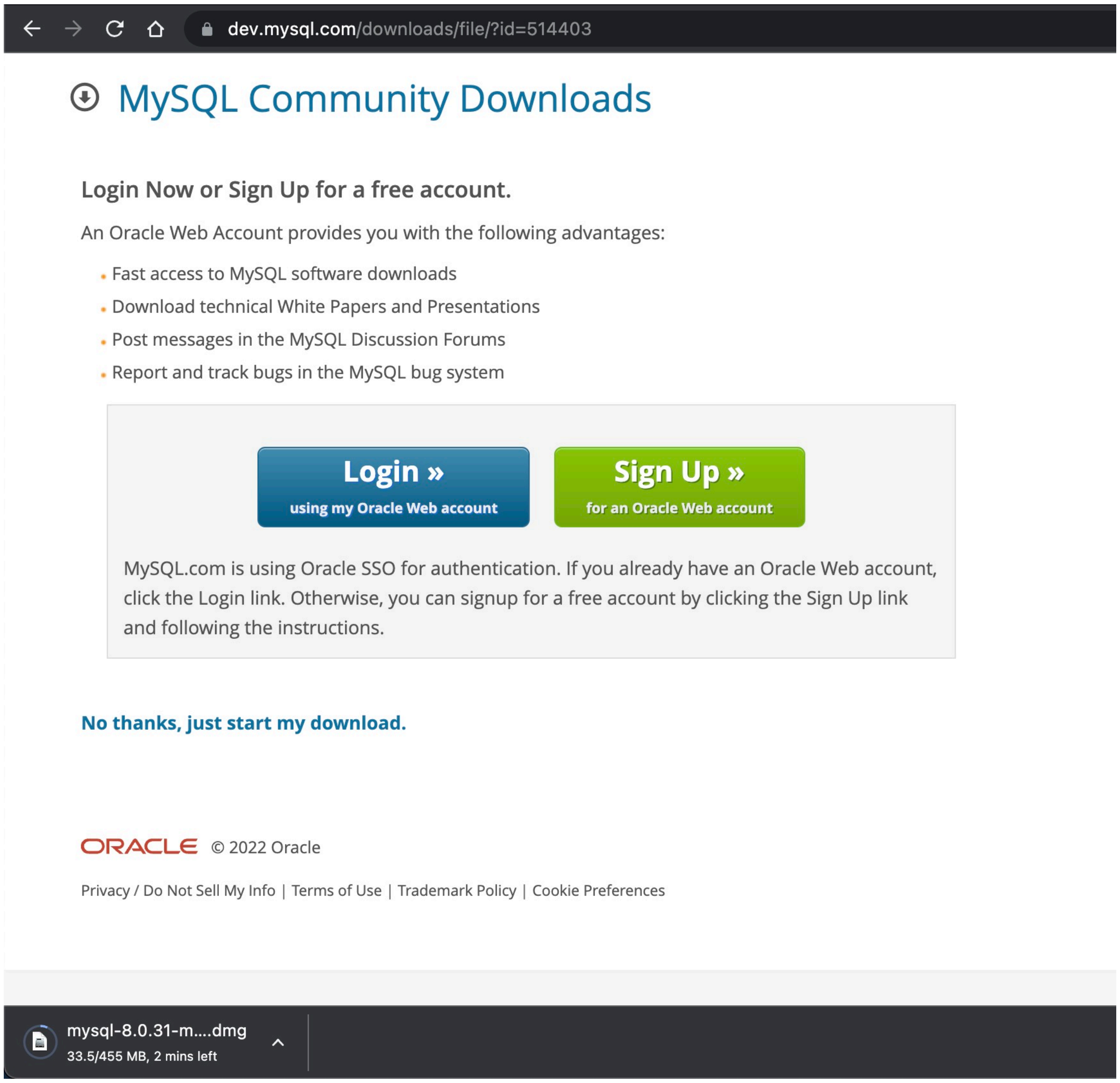
Select OS Version: All

Looking for previous GA versions?

Packages for Monterey (12) are compatible with Ventura (13) and Big Sur (11)

macOS 12 (ARM, 64-bit), DMG Archive	8.0.31	454.9M	Download
(mysql-8.0.31-macos12-arm64.dmg)		MD5: 1beba0f368a30dd7cf9eb0ef4a816a4e | Signature	
macOS 12 (x86, 64-bit), DMG Archive	8.0.31	461.8M	Download
(mysql-8.0.31-macos12-x86_64.dmg)		MD5: 4a6f2c7447f9b2895a2f5710f0178904 | Signature	

2. Once the download is complete, open "mysql-8.0.31-macos12-arm64.dmg". This file can be found either in the downloads toolbar at the bottom your browser window on Google Chrome, or in the "Downloads" folder of your computer. In the subsequent window, click on the icon for "mysql-8.0.31-macos12-arm64.pkg" and allow the package to run its program.



dev.mysql.com/downloads/file/?id=514403

MySQL Community Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

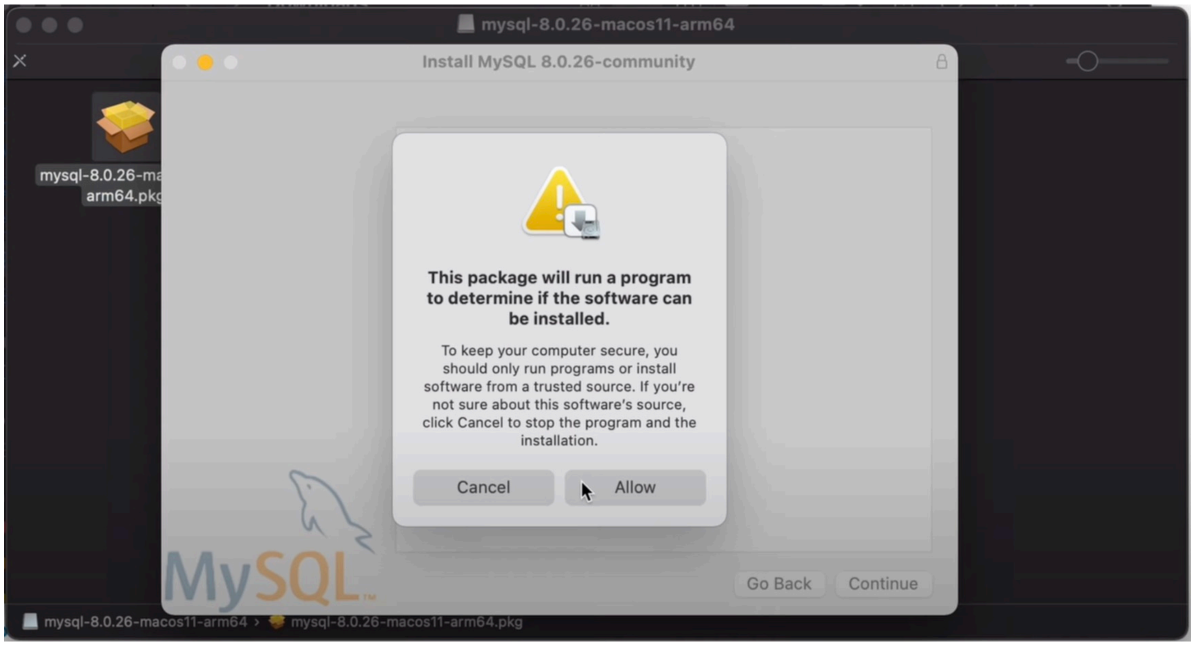
MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.

ORACLE © 2022 Oracle

Privacy / Do Not Sell My Info | Terms of Use | Trademark Policy | Cookie Preferences

mysql-8.0.31-m....dmg
33.5/455 MB, 2 mins left

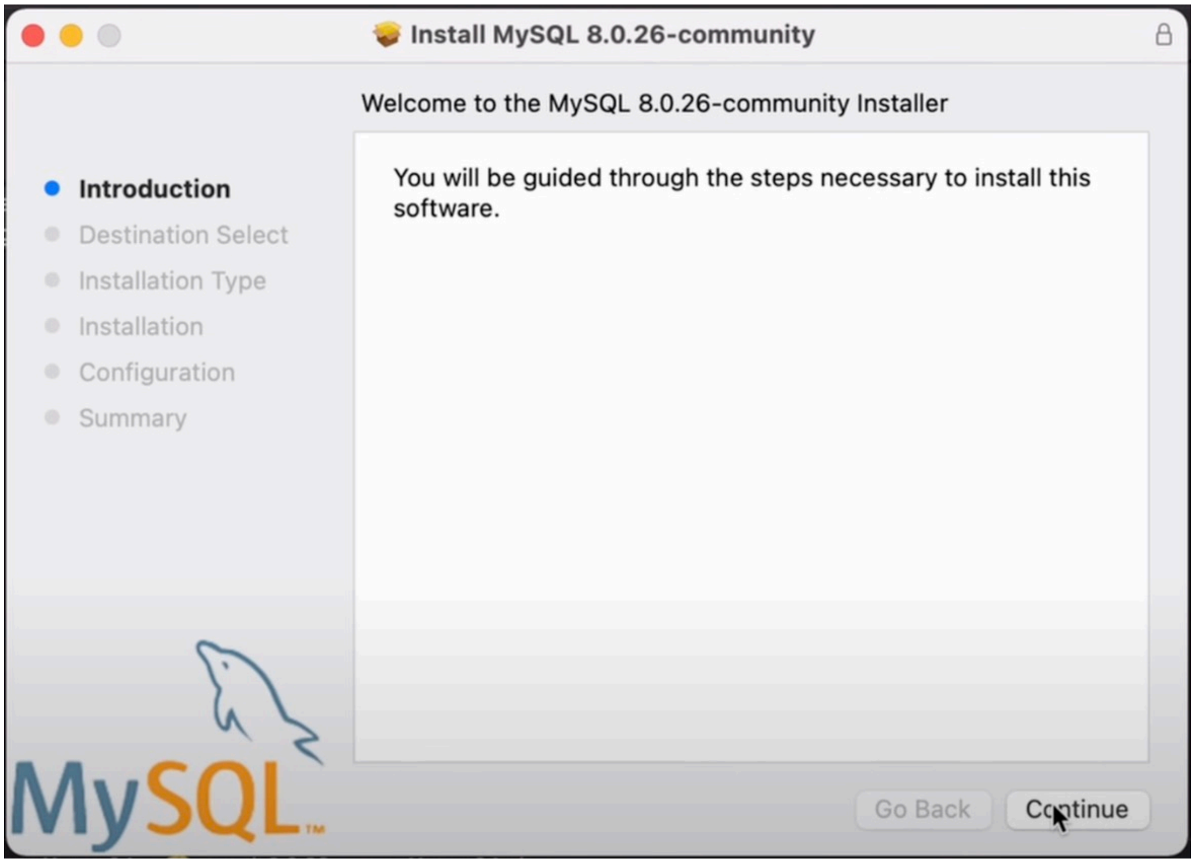


This package will run a program to determine if the software can be installed.

To keep your computer secure, you should only run programs or install software from a trusted source. If you're not sure about this software's source, click Cancel to stop the program and the installation.

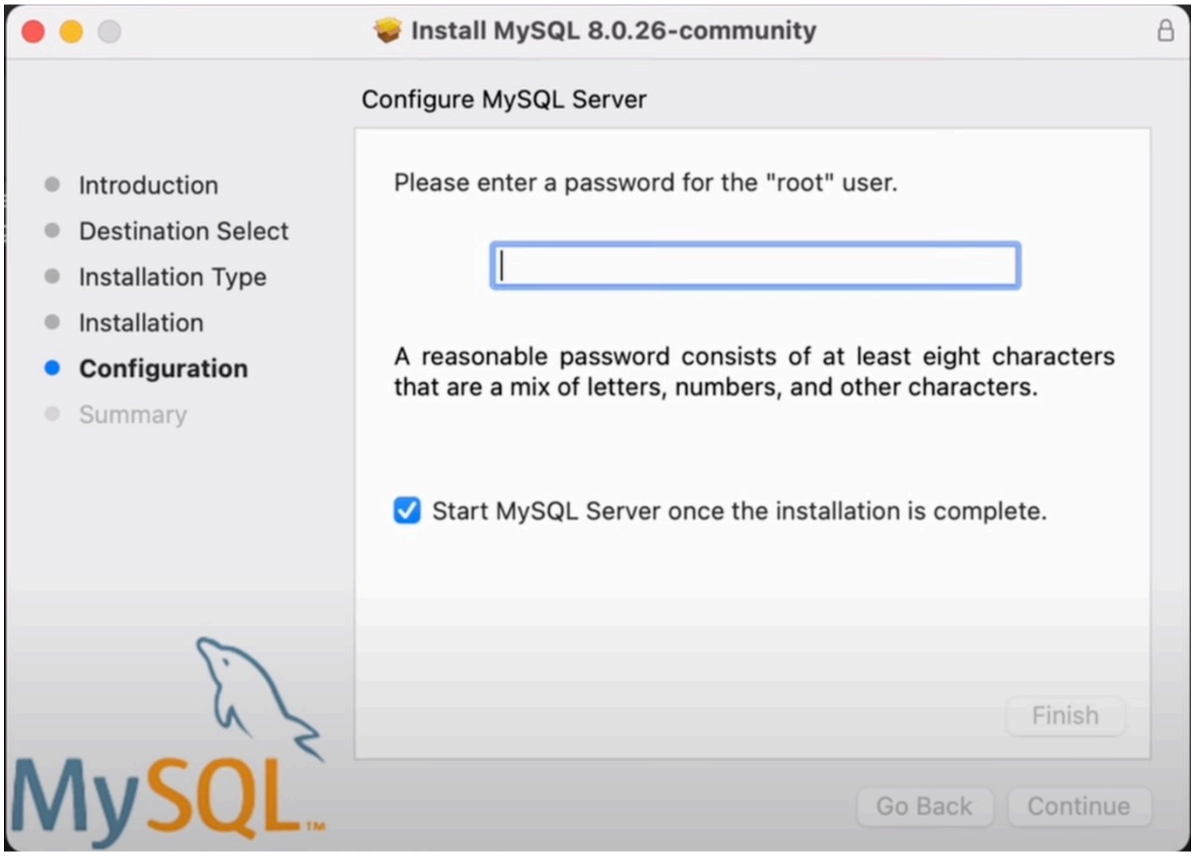
mysql-8.0.26-macos11-arm64
Install MySQL 8.0.26-community
mysql-8.0.26-ma arm64.pkg
Cancel
Allow
Go Back
Continue
mysql-8.0.26-macos11-arm64 › mysql-8.0.26-macos11-arm64.pkg

3. Follow the installation steps in the "Install MySQL 8.0.31 Community" window. At the Configuration step in the sidebar below, the installer will prompt you to assign a password to an initial "root" account. Remember this password for configuring your database later. When the installation has completed successfully, you may move the installer to Trash.



Welcome to the MySQL 8.0.26-community Installer

You will be guided through the steps necessary to install this software.


Install MySQL 8.0.26-community

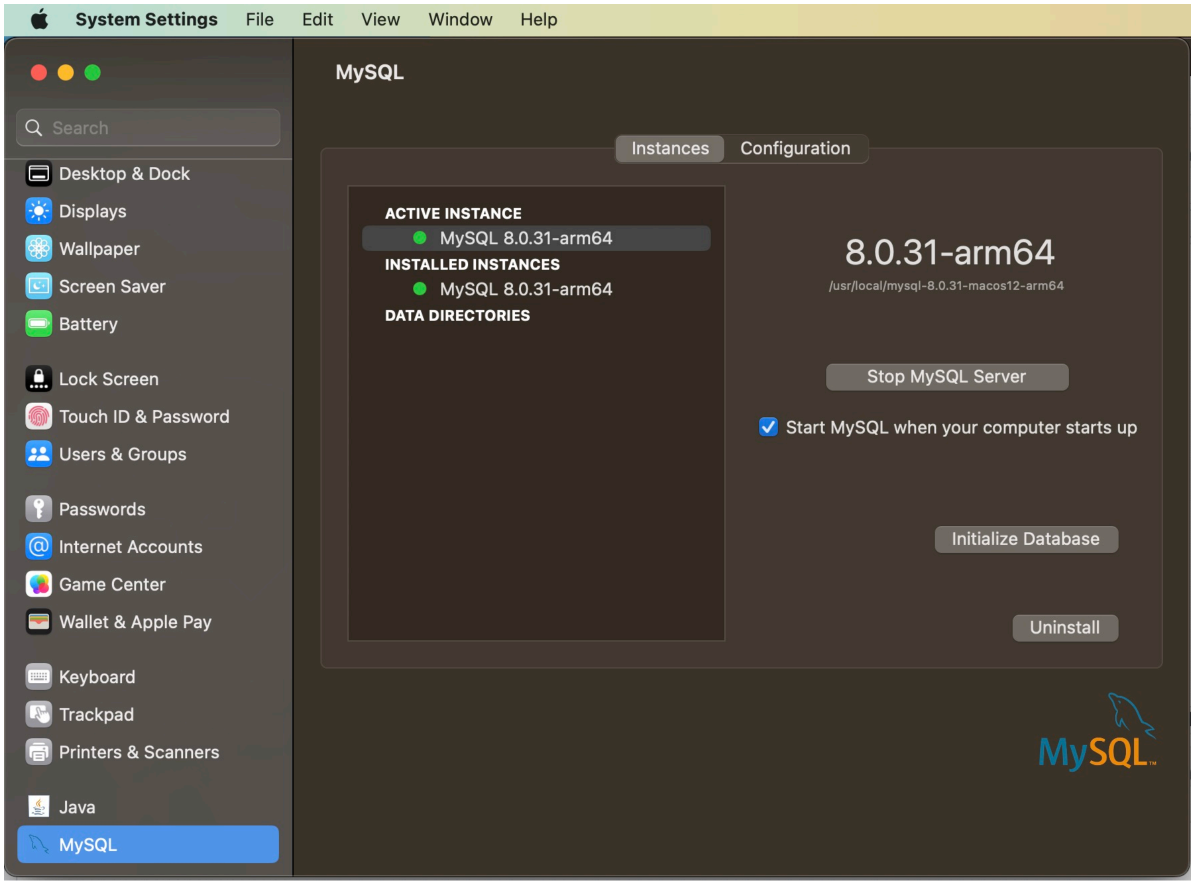
Configure MySQL Server

Please enter a password for the "root" user.

A reasonable password consists of at least eight characters that are a mix of letters, numbers, and other characters.

☑ Start MySQL Server once the installation is complete.

4. To check the installation, open System Settings a.k.a System Preferences and click on the MySQL icon at the bottom of the navigation bar. You can see whether the MySQL server is running.



MySQL

ACTIVE INSTANCE
● MySQL 8.0.31-arm64
INSTALLED INSTANCES
● MySQL 8.0.31-arm64
DATA DIRECTORIES

8.0.31-arm64
/usr/local/mysql-8.0.31-macos12-arm64

[✓] Start MySQL when your computer starts up

5. After MySQL has been installed this way, you may still have to configure the path to the MySQL server. In other words, you want to be able to start up the MySQL shell from the command line by running `$ mysql` (don't type the `$`). When you open terminal, your default interactive shell automatically executes a hidden configuration file in your home directory that contains commands to do a number of things upon log-in, such as adding a directory to `$PATH`, exporting some variable, and customizing the shell session. The file is called bash_profile for Bash, .zprofile for Z shell (zsh), and et cetera for related shells. So for MacOS, you will add `export PATH=$PATH:/usr/local/<your_mysql_folder>/bin` to the end of the .bash_profile file so that your shell can find the path to MySQL on your machine and open a MySQL shell.

- `export` is a built-in command that sets an environment variable to be exported to child-processes (or subshells), so that the child inherits them. For example:

```
$ foo=bar
$ bash -c 'echo $foo'

$ export foo
$ bash -c 'echo $foo' bar
```

- `PATH` is an environment variable specifying a set of directories where executable programs are located.
- `${PATH}` is the default `PATH` assignment.
- The combination `PATH="${PATH}:<some_path_to_a_directory>"` extends the `PATH` variable to the desired directory. In Bash, this is a colon (:) separated list.

First, find the path to MySQL. On MacOS, the MySQL directories are installed under `/usr/local/` by default. Navigate to `/usr/local/` using Spotlight (Command-Space and enter into the search bar) and open the "mysql" folder e.g. "mysql-8.0.31-macos12-arm64". Right click on the bin folder, hold the Option key, and click on "Copy 'bin' as Pathname".

Then, run the following commands in Terminal to open .bash_profile.

```
$ ls -a                 # list all the files in your home directory with
                          # the -a option to include  hidden files.
$ touch .bash_profile   # if you don't see it listed, you can create one
$ ls -a                 # check that it's there now
$ open -t .bash_profile # open the text file. The -t option will open it with
                          # your default text editor.
```

Lastly, add `export PATH=$PATH:<paste_your_mysql_path_here>` e.g. `export PATH=$PATH:/usr/local/mysq` to the end of .bash_profile. To save and exit, you should just be able to Command-S to save and close the window.

6. Now, you should be able to connect to MySQL from the command line. The `$ mysql` command invokes mysql without specifying any explicit connection parameters. Because there are no parameter options, the default values apply:

- The default host name is localhost.
- The default user name is ODBC on Windows, your Unix login name on Unix, and 'root' on MacOS.
- No password is sent because neither –password nor -p is given. If all your users have passwords, you may get an error: `ERROR 1045 (28000): Access denied for user 'USERNAME'@'localhost' (using password: NO)`
- The first non-option argument is taken as the name of the default database. Because there is no such argument, MySQL selects no default database.

7. To specify the host name and user name explicitly, as well as a password, supply appropriate options on the command line. To select a default database, add a database-name argument. You can use either of these commands below:

```
mysql -host=HOSTNAMEorIP -user=USERNAME -password=PASSWORD DATABASENAME
mysql -h HOSTNAMEorIP -u USERNAME -pPASSWORD DATABASENAME
```

The options above mean:

```
--host or -h: host machine or IP address running the server
--user or -u: username
--password or -p: password (**no space between -p and the password text**). To avoid security
- the name of the database that you want to connect.
```

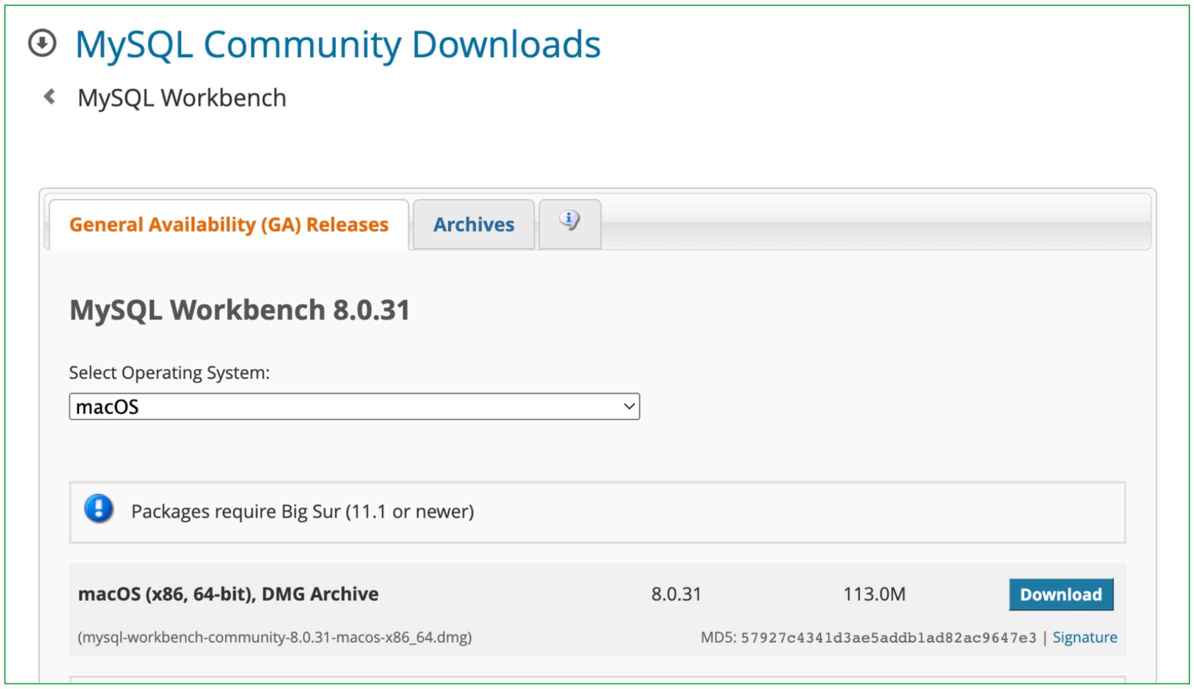
For the example in this vignette, you can try using the MySQL command line to see that 'bikes' is in the list of all databases, and then exit to your default terminal.

```
$ mysql -u root -p
Enter password: <YOUR ROOT PASSWORD>
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 161
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| bikes              |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.06 sec)

mysql> exit
Bye
```

Installing MySQL Workbench

Optionally, you can install MySQL Workbench, which is a unified visual tool for working with MySQL servers and databases. This is like what RStudio is for R, if you plan on using more SQL in the future.

1. Download MySQL Workbench 8.0.31. Be sure to select the correct operating system and processor for your computer. For Macbook M1/M2, please select "macOS (x86, 64-bit), DMG Archive". You can start the download without creating an Oracle Web Account.


MySQL Community Downloads

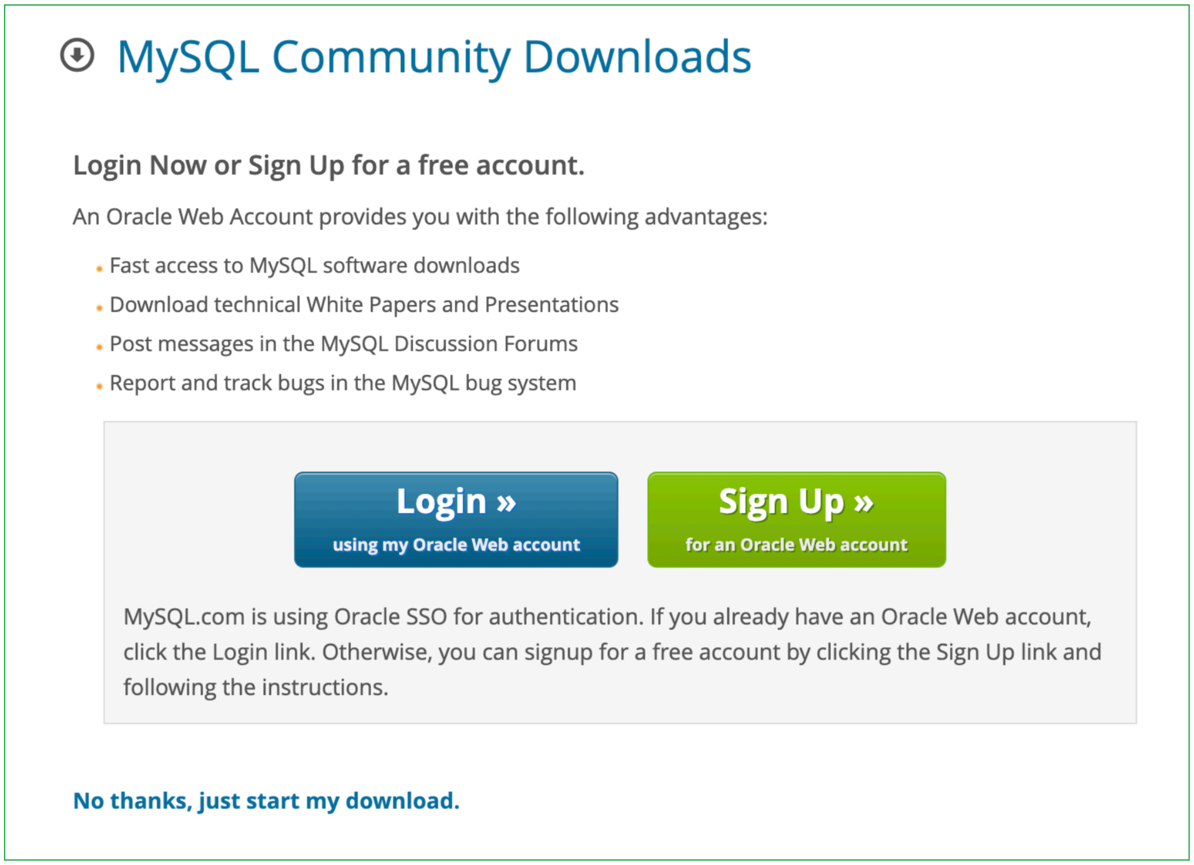
MySQL Workbench

MySQL Workbench 8.0.31

Select Operating System: macOS

Packages require Big Sur (11.1 or newer)

macOS (x86, 64-bit), DMG Archive	8.0.31	113.0M	Download
(mysql-workbench-community-8.0.31-macos-x86_64.dmg)			MD5: 57927c4341d3ae5addb1ad82ac9647e3 | Signature



MySQL Community Downloads

Login Now or Sign Up for a free account.

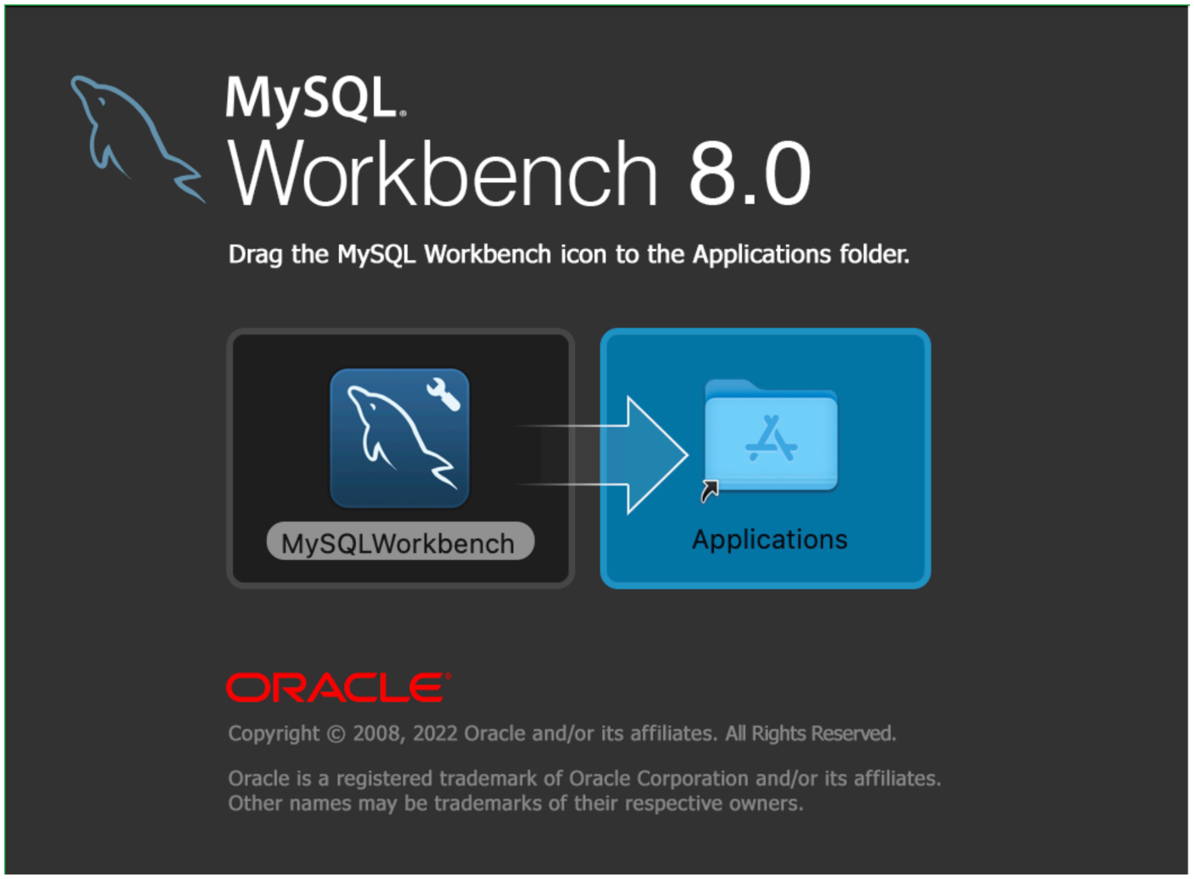
An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

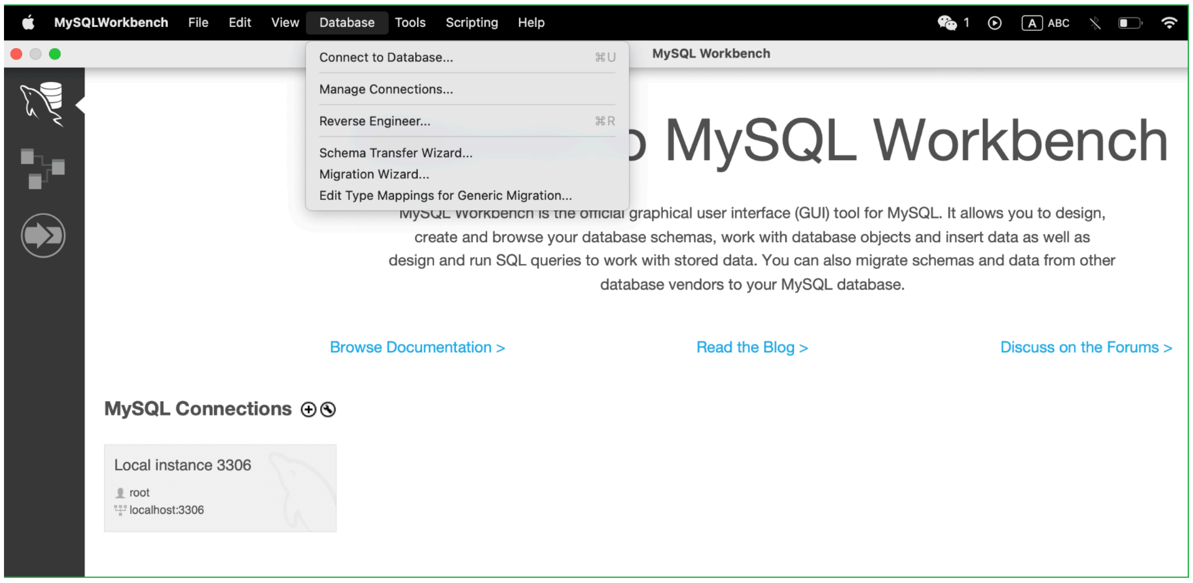
MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.

2. Once the download is complete, open "mysql-workbench-community-8.0.31-macos-x86_64". This file can be found either in the downloads toolbar at the bottom your browser window (Google Chrome) or in the "Downloads" folder of your computer. In the subsequent window, click on the icon for "mysql-8.0.31-macos12-arm64.pkg" and allow the package to run its program. Then, you can drag MySQLWorkbench into the Applications Icon to complete the installation.

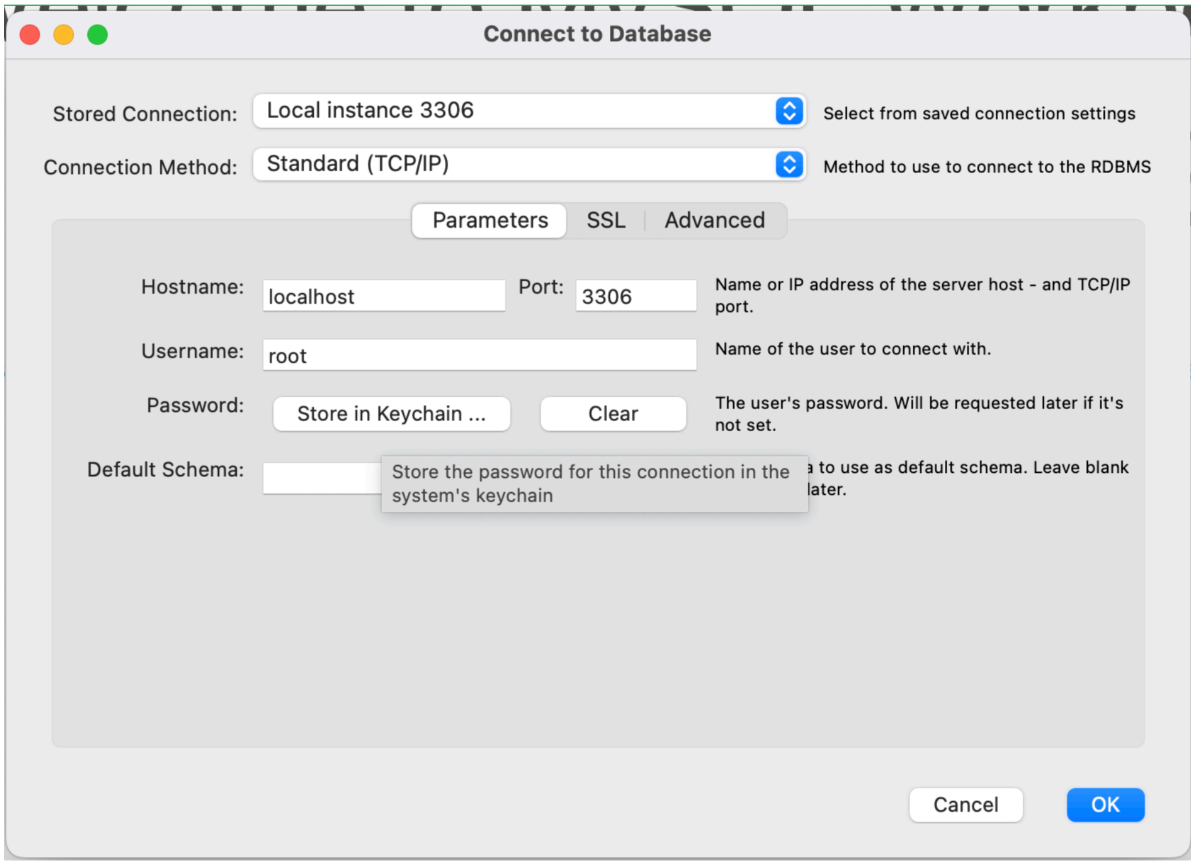

MySQL
Workbench 8.0
Drag the MySQL Workbench icon to the Applications folder.
MySQLWorkbench
Applications
ORACLE
Copyright © 2008, 2022 Oracle and/or its affiliates. All Rights Reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

3. Once the installation is complete, open the MySQLWorkbench. Click the database button at the upper left corner and then click the “Connect to Database”.



MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

4. Then, you need to click "store in Keychain" and enter the password you set when you download the MySQL server. Now, you successfully connect to the database from MySQLWorkbench.



Connect to Database

Stored Connection: Local instance 3306 — Select from saved connection settings

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

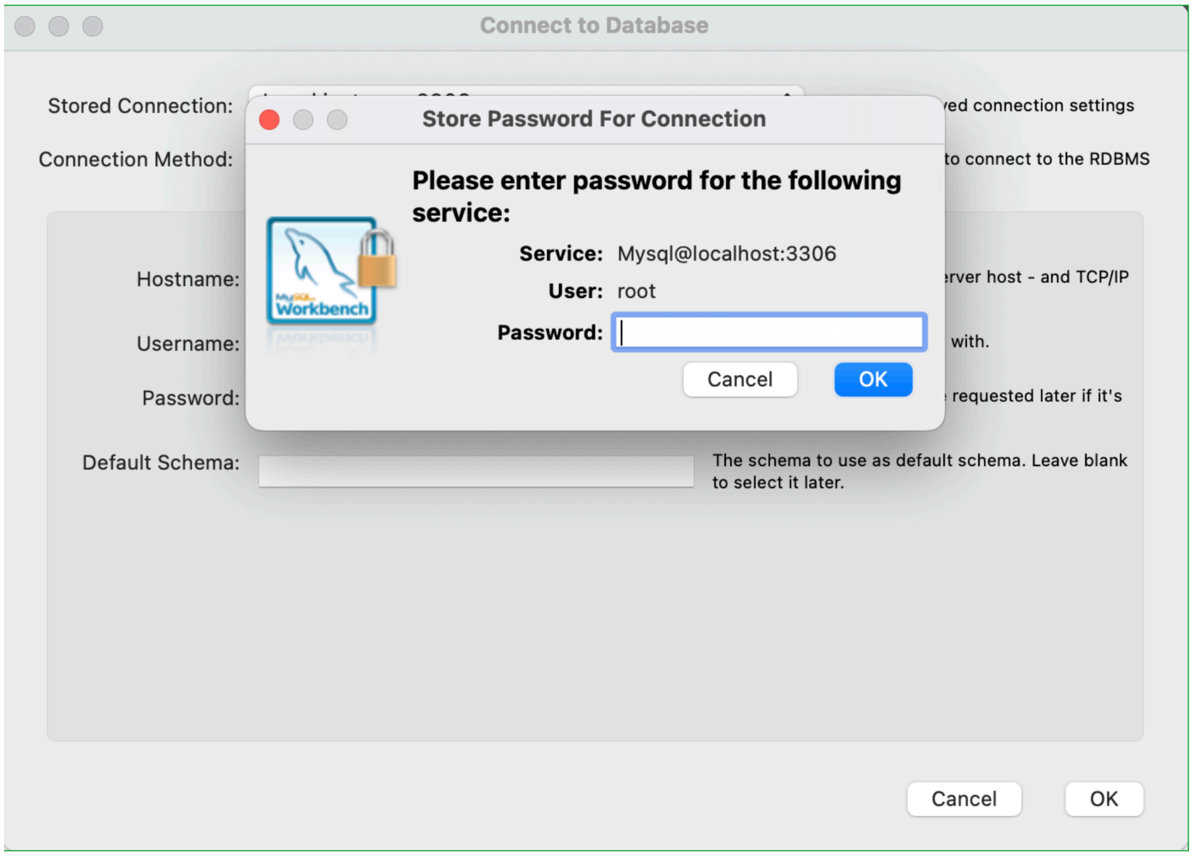
Hostname: localhost Port: 3306 — Name or IP address of the server host - and TCP/IP port.

Username: root — Name of the user to connect with.

Password: Store in Keychain ... Clear — The user's password. Will be requested later if it's not set.

Default Schema: a to use as default schema. Leave blank later.

Store the password for this connection in the system's keychain



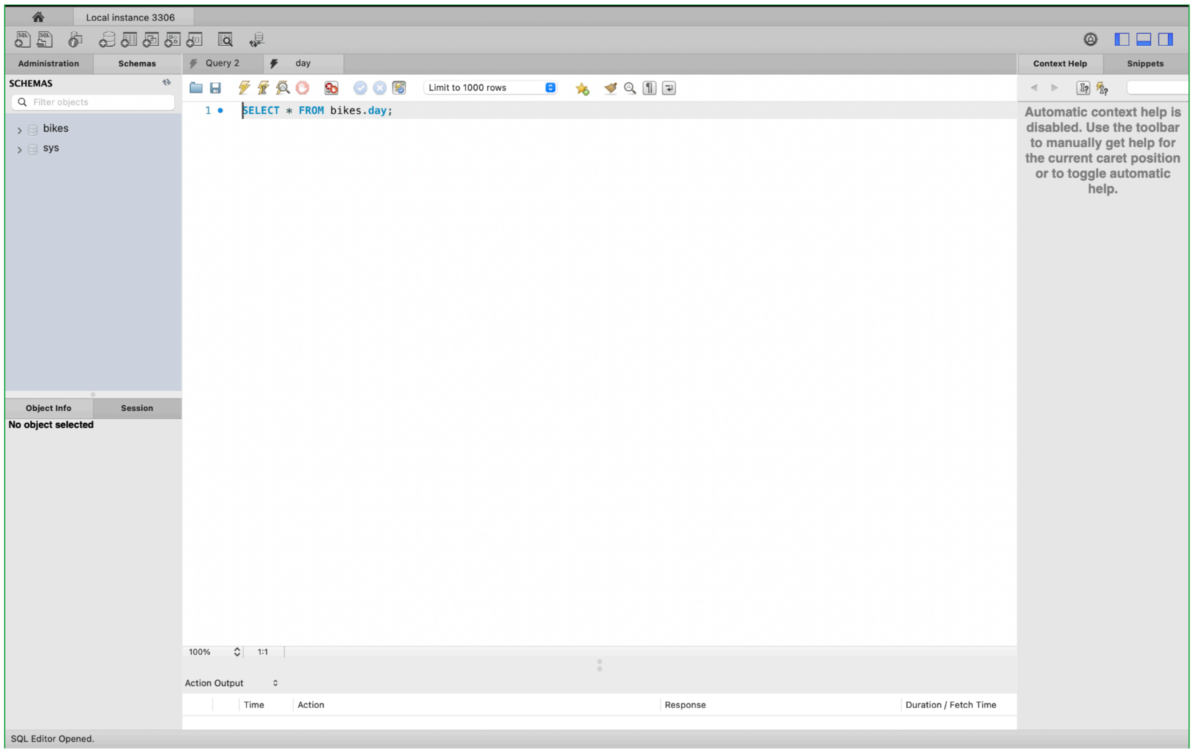
Please enter password for the following service:

Service: Mysql@localhost:3306

User: root

Password:

The schema to use as default schema. Leave blank to select it later.



```sql
SELECT * FROM bikes.day;
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

SQL Editor Opened.

Data Introduction

The sample data used to demonstrate the methods in this vignette is the Bike Sharing Dataset from the University of California, Irvine Machine Learning Center. This dataset contains the hourly and daily count of rental bikes between years 2011 and 2012 in the Capital bikeshare system with the corresponding weather and seasonal information. It can be found in our 'data' directory, which contains two .csv files and a Readme with more data description.

Now, we can get started in R.

Preliminary Steps

1. Install and load the required package.

```r
install.packages("RMariaDB")
library(RMariaDB)
```

2. Import .csv files as data frames.

```r
df_day <- read.csv("data/day.csv", header=TRUE)
head(df_day)

df_hour <-read.csv("data/hour.csv", header=TRUE)
head(df_hour)
```

3. Build the Database.

```r
# connect R to MySQL by creating a MySQL connection object
con <- dbConnect(RMariaDB::MariaDB(),
                 user = 'root',          # YOUR USERNAME. Here, we use the default.
                 password = 'password', # YOUR PASSWORD HERE.
                 host = 'localhost')    # A string identifying the host machine running th
                                        # server. Here, we use the default.

# create a new database
dbSendQuery(con, "CREATE DATABASE bikes")
dbSendQuery(con, "USE bikes")

# reconnect to the database that we just created
mydb <- dbConnect(RMariaDB::MariaDB(),
                 user = 'root',
                 password = 'password',
                 host = 'localhost',
                 dbname = 'bikes')     #

# Write the data frames to database tables
dbWriteTable(conn = mydb,    # name of your database connection object
             name = "day",   # set a name for your table here
             value = df_day) # specify the dataframe to get the data from

dbWriteTable(conn = mydb,
             name = "hour",
             value = df_hour)

# list the tables available in bikes
dbListTables(mydb)

# list the fields in a table
dbListFields(mydb, "day")
```

```
# view the whole table
dbReadTable(mydb, "day")
```

Queries

Basic Syntax

Function: SELECT

Purpose: It is used to select data based on specific conditions

```
# we want to select day.season and day.yr from day. we can also select other columns from
dbGetQuery(mydb, 'SELECT day.season, day.yr FROM day')
```

Function: INSERT

Purpose: It is used to insert data into the tables with specific values

```
# we want to insert new values 1 and 0 into day.season and day.yr
dbGetQuery(mydb, 'INSERT INTO day(day.season, day.yr)
                  VALUES (1,0)')
```

Function: UPDATE

Purpose: It is used to update data into the tables with specific values

```
# we want to update weekday to 9 where workingday equals to 0 in day. we can also use mult
dbGetQuery(mydb, 'UPDATE day
                  SET weekday=8
                  WHERE workingday=0')
```

Function: DELETE

Purpose: It is used to delete specific data in a table

```
# we want to delete values from day where workingday equals to 0. we can also set other co
dbGetQuery(mydb, 'DELETE FROM day
                  WHERE workingday=0')
```

Filtering Clause:

Function: WHERE

Purpose: It is used to filter records based on a specific condition

```
# Selecting an instant from the day database wehre the temp is 0.3441670 and limit it to 2
dbGetQuery(mydb, "SELECT instant FROM day where temp = 0.3441670 LIMIT 2" )
```

Function: ORDER BY

Purpose: It is used to sort the result-set in ascending or descending order.

```
# Selecting the temperature from the day database and ordering our ouput by season in Asce
dbGetQuery(mydb, "SELECT temp FROM day ORDER BY season ASC LIMIT 3" ) # Ascending Order
# Selecting the temperature from the day database and ordering our ouput by season in Desc
dbGetQuery(mydb, "SELECT temp FROM day ORDER BY season DESC LIMIT 3" ) # Descending Order
```

Function: LIMIT

Purpose: It is used to select a limited number of records.

```
#Select the date from the day database where the tempeature is greater than 0 and limiting
dbGetQuery(mydb, "SELECT dteday FROM day where temp > 0 LIMIT 5" )
```

Function: LEFT JOIN

Purpose: Left Join returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

```
# Selecting windspeed from the day database and casual from the hour database and left joi
dbGetQuery(mydb, 'SELECT day.windspeed, hour.casual FROM day LEFT JOIN hour ON day.season
```

Function: RIGHT JOIN

Purpose: Right Join returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

```
# Selecting windspeed from the day database and casual from the hour database and right jo
dbGetQuery(mydb, 'SELECT day.windspeed, hour.casual FROM day RIGHT JOIN hour ON day.season
```

Function: INNER JOIN

Purpose: Inner Join selects records that have matching values in both tables.

```
# Selecting windspeed from the day database and casual from the hour database and inner jo
dbGetQuery(mydb, 'SELECT day.windspeed, hour.casual FROM day INNER JOIN hour ON day.season
```

Function: OUTER JOIN/ FULL JOIN

Purpose: Outer Join keyword returns all records when there is a match in left (table1) or right (table2) table records. MySQL does not support the full join so we use the following way to mimic the same process.

```
dbGetQuery(mydb, "SELECT day.windspeed, hour.casual FROM day LEFT JOIN hour ON day.season
           SELECT day.windspeed, hour.casual FROM day RIGHT JOIN hour ON day.season = hour
```

Function: UNION

Purpose: We use the union operator to combine the result-set of two or more SELECT statements.

```
# We are selecting cnt from the day database and also selecting temperature from the hour
dbGetQuery(mydb, "SELECT day.cnt FROM day UNION SELECT temp FROM hour LIMIT 2")
```

Function: IN

Purpose: The in operator allows you to specify multiple values in a where clause or shorthand for multiple or conditions

```
# Selecting attributes from the day database where season in the select season from the da
dbGetQuery(mydb, "SELECT* FROM day WHERE season IN (SELECT season FROM day) LIMIT 5")
```

Function: LIKE

Purpose: The like operator is used in a where clause to search for a specified pattern in a column. We use the percent operator (%) to represent 0, 1, or multiple characters. The underscore (_) represents a single character.

```
# Selecting temp and season from the day database where the date... and limit to 4 outputs
dbGetQuery(mydb, "SELECT temp,season FROM day WHERE dteday LIKE '%2' LIMIT 4") # End in 2
dbGetQuery(mydb, "SELECT temp,season FROM day WHERE dteday LIKE '2%' LIMIT 4") # Start wit
dbGetQuery(mydb, "SELECT temp,season FROM day WHERE dteday LIKE '%2%' LIMIT 4") # Has 2 in
dbGetQuery(mydb, "SELECT temp,season FROM day WHERE dteday LIKE '2%2' LIMIT 4") # Start wi
```

Function: BETWEEN

Purpose: The between operator selects values within a given range. The values can be numbers, text, or dates.

```
#Select the date and atemp from day where the temp is between 0 and 1 and limit to 5 outpu
dbGetQuery(mydb, "SELECT dteday, atemp FROM day where temp BETWEEN 0 AND 1 LIMIT 5" )
```

Function: GROUP BY

Purpose: The group by statement groups rows that have the same values into summary rows

```
# Select the number of season from day database and group by date the output and order by
dbGetQuery(mydb, "SELECT COUNT(season) FROM day GROUP BY dteday ORDER BY COUNT(season) DES
```

Function: HAVING

Purpose: The having clause was added to SQL because the where keyword cannot be used with aggregate functions.

```
# Select the date from the day database and group by date having the count of the temperat
dbGetQuery(mydb, "SELECT dteday FROM day GROUP BY dteday HAVING COUNT(temp) > 0.2 LIMIT 5"
```

Function: CASE

Purpose: expression goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the else clause.

```
# Select windspeed and when the windspeed is less than 0.1 output low else Null
dbGetQuery(mydb, "SELECT windspeed, CASE
            WHEN windspeed < 0.1 THEN 'LOW'
            ELSE NULL
            END AS CHECK_LOWSPEED FROM day")
```

Function: DISTINCT

Purpose: The distinct statement is used to return only distinct (different) values

```
# Select distinct date from day where the temp is 0.3441670
dbGetQuery(mydb, "SELECT DISTINCT(dteday) FROM day where temp = 0.3441670" )
dbGetQuery(mydb, "SELECT DISTINCT(windspeed) FROM day where temp = 0.1" ) # Value Does not
```

Numerical Functions

Function: RAND

Purpose: Generate a series of random numbers

```
# We can see that a random number from 0 to 1 is chosen.
dbGetQuery(mydb, "SELECT RAND()" )
```

Function: ROUND

Purpose: The ROUND() function rounds a number to a specified number of decima places. For example, we want to round up temperature and windspeed to 3 decimals

```
# We can see that temperature and windspeed are round from 7decimals to 3 decimals.
dbGetQuery(mydb, "SELECT temp,round(temp,3),windspeed, round(windspeed,
3) FROM day" )
```

Function: FLOOR

Purpose: The FLOOR() function returns the largest integer value that is smaller than or equal to a number. We want to see the largest integer value which is either equal to or less than the given average or minimum windspeed.

```
# Given the average value of windspeed and minimum value of windspeed, we see largest inte

dbGetQuery(mydb,"SELECT AVG(windspeed),MIN(windspeed),FLOOR(AVG(windspeed)) AS
Lower_Average FROM day" )
dbGetQuery(mydb, "SELECT
AVG(windspeed),MIN(windspeed),FLOOR(MIN(windspeed)) AS Lower_Average
FROM day" )
```

Function: CEIL

Purpose: The CEIL() function returns the smallest integer value that is bigger than or equal to a number. It is equal to the CEILING() function. we want to see the smallest integer value which is either greater than or equal to the given average or minimum number.

```
dbGetQuery(mydb, "SELECT
AVG(windspeed),MIN(windspeed),CEIL(AVG(windspeed)) AS Lower_Average FROM
day" )
dbGetQuery(mydb, "SELECT
AVG(windspeed),MIN(windspeed),CEIL(MIN(windspeed)) AS Lower_Average FROM
day" )
```

Function: Truncate(n,d) with positive d

Purpose: we want to see atemp keeps 2 decimals and hum only keeps 1 decimals

```
dbGetQuery(mydb, "SELECT atemp, hum,
TRUNCATE(atemp,2), TRUNCATE(hum,1) FROM day" )
```

Function: #Truncate(n,d) with negative d we want to see atem and hum are truncated

Purpose: we want to see two or one digit left to the decimal points.

```
dbGetQuery(mydb,
"SELECT atemp, hum, TRUNCATE(atemp,-2), TRUNCATE(hum,-1) FROM day" )
```

Function: COUNT ALL

Purpose: How many row we have in the dataset

```
dbGetQuery(mydb, "SELECT COUNT(*) FROM day" )
```

Function: COUNT(Distinct())

Purpose: How many unique value (weekday) we have in the dataset

```
dbGetQuery(mydb, "SELECT COUNT(DISTINCT (weekday)) FROM day" )
```

Function: avg()

Purpose: the average value of a target variable

```
# we want to see the average speed
dbGetQuery(mydb, "SELECT AVG(windspeed) AS AVERAGE_SPEED FROM day" )
```

Function: sum()

Purpose: the sum value of a target variable

```
# we want to see the sum of casual users
dbGetQuery(mydb, "SELECT SUM(casual) AS TOTAL_NUM_CASUAL_USER FROM day")
```

Function: max()

Purpose: the max value of a target variable

```
# we want to see the max speed of wind
dbGetQuery(mydb, "SELECT MAX(windspeed) AS MAX_WINDSPEED FROM day")
```

Function: min()

Purpose: the min value of a target variable

```
# we want to see the min speed of wind
dbGetQuery(mydb, "SELECT MIN(windspeed) AS MIN_WINDSPEED FROM day")
```

Function: concat()

Purpose: concatenate different string to one string

```
# We want to concat days and hours as one column

dbGetQuery(mydb, "SELECT CONCAT(dteday,' Hour: ',hr) FROM hour")
```

Function: length()

Purpose: to see how many characters composes a specific string

```
# We want to see how many characters composes the date
dbGetQuery(mydb, "SELECT DISTINCT(LENGTH(dteday)) AS length_of_char_date FROM hour")
```

Function: lower()

Purpose: to make all characters to lower case

```
#WE want to firstly create windspeed breakdown to two part low and high. Then we want to
# show how to use lower function to make all spring be lower case
dbGetQuery(mydb, "SELECT LOWER(SPEED_BREAKDOWN) FROM (SELECT windspeed, CASE
            WHEN windspeed < 0.1 THEN 'LOW'
            ELSE 'HIGH'
            END AS SPEED_BREAKDOWN FROM day) AS tmp")
```

Function: upper()

Purpose: to make all characters to upper case

```
#  We want to show how to use upper function to make all spring be upper case
dbGetQuery(mydb, "SELECT UPPER(SPEED_BREAKDOWN) FROM (SELECT windspeed, CASE
            WHEN windspeed < 0.1 THEN 'low'
            ELSE 'high'
            END AS SPEED_BREAKDOWN FROM day) AS tmp")
```

dbplyr

We can also use dbplyr to query the data.

```
library(dbplyr)
library(dplyr)

# CONNECT TO DATABASE
con <- dbConnect(RMariaDB::MariaDB(),
                  user = 'root',
                  password = 'password', # YOUR PASSWORD HERE
                  host = 'localhost',
                  dbname = 'bikes')

# CONNECT TO TABLE
data_sample<- con %>% tbl("day")

# SELECT 4 columns. In SQL we have to write down the four columns' name separately, but in
data_sample %>% select(season:holiday)

# We can use filter to substitute where condition in SQL. Here we are showing data that ha
data_sample %>% filter(windspeed > 0.4)

# We can also do the same SQL group by and aggregation task by uing dbplyr group_by functi
data_sample %>% group_by(weekday) %>%
  summarise(
    count_day = count(instant)
  )
```

However if we want to realize join function, we need to use show_query() function to identify the join key. In this case, we still need to use SQL syntax in order to do the complicated correct join.

In summary, dbplyr can be used in simple aggregation tasks without SQL knowledge. However, we still need to know SQL in order to finish more complicated tasks, especially with large data.

Works Cited:

1. MySQL Installation Guide: https://dev.mysql.com/doc/mysql-installation-excerpt/5.7/en/
2. MySQL Server Configuration Guide: https://www.youtube.com/watch?v=nj3nBCwZaqI
3. MySQL Workbench Installation Guide: https://www.youtube.com/watch?v=6FvvWhiZyDY
4. Sample data: https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset

5. More on RMariaDB: https://rdrr.io/cran/RMariaDB/, https://rmariadb.r-dbi.org/reference/dbconnect-mariadbdriver-method
6. More on dbplyr and JOIN: https://dbplyr.tidyverse.org/reference/join.tbl_sql.html
7. Colburn, Rafe. Special Edition Using SQL. Que, 2000.
8. Röhm, Uwe, et al. "SQL for Data Scientists: Designing SQL Tutorials for Scalable Online Teaching." Proceedings of the VLDB Endowment, vol. 13, no. 12, 2020, pp. 2989–92, https://doi.org/10.14778/3415478.3415526.
9. Guo, Aibo, et al. "ER-SQL: Learning Enhanced Representation for Text-to-SQL Using Table Contents." Neurocomputing (Amsterdam), vol. 465, 2021, pp. 359–70, https://doi.org/10.1016/j.neucom.2021.08.134.
10. Guo, Aibo, et al. "ER-SQL: Learning Enhanced Representation for Text-to-SQL Using Table Contents." Neurocomputing (Amsterdam), vol. 465, 2021, pp. 359–70, https://doi.org/10.1016/j.neucom.2021.08.134.
11. Reese, George., et al. Managing and Using MySQL. 2nd ed., O'Reilly, 2002.