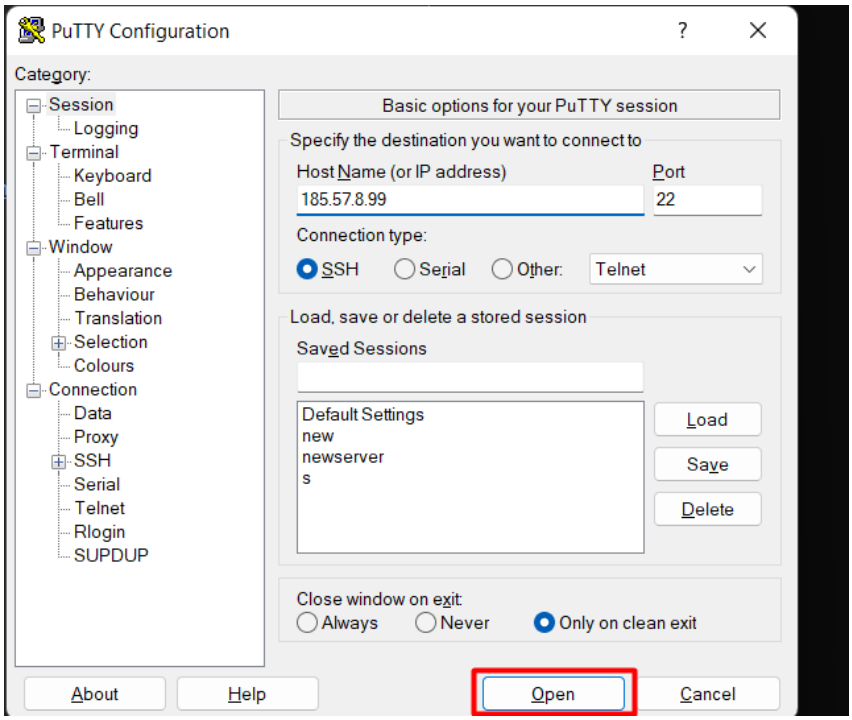
SEAMLess Database Setup Guide

Server Setup

First, you will need to download and install Putty in order to connect to the virtual Linux machine: https://www.putty.org/

You can open a connection by entering the IP address provided by your web hosting service into Host Address and clicking open.



You will now see a terminal window where you can log in. If you have a new server you will need to use the default root user and password provided to you by the web hosting service.

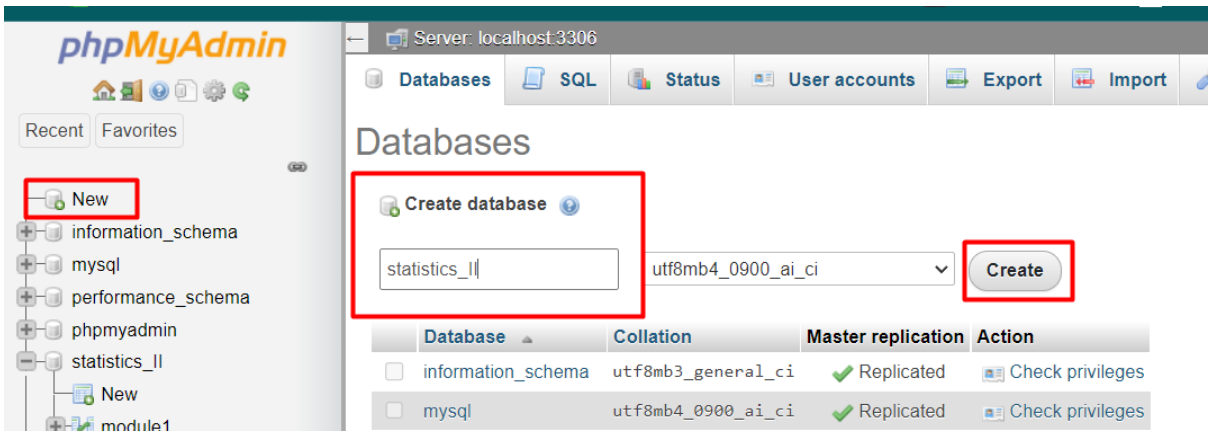
You will need to use the terminal to install LAMP stack and phpMyAdmin, follow the instructions here: https://www.linuxtechi.com/install-phpmyadmin-linux/

The database settings can be configured manually in the file my.cnf file usually located at /etc/mysql/my.cnf, which you can open with nano. It is a text file where you change the values and save the file.

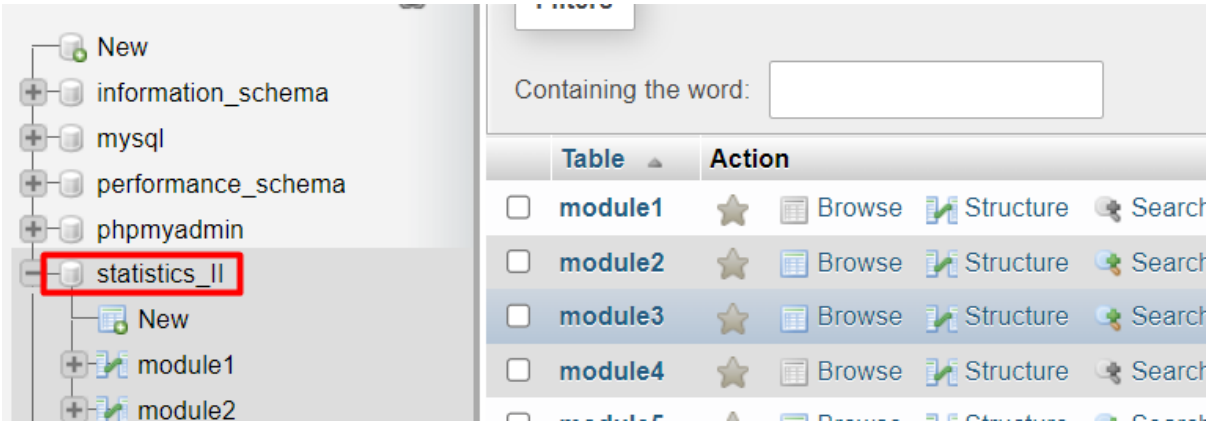
Database Setup

The easiest way to do this is:

1. Go to phpMyAdmin (your_server_url/phpmyadmin/). The server url should be provided by your hosting service.
2. Create a new database, give it the name of the course



3. Navigate to the new database

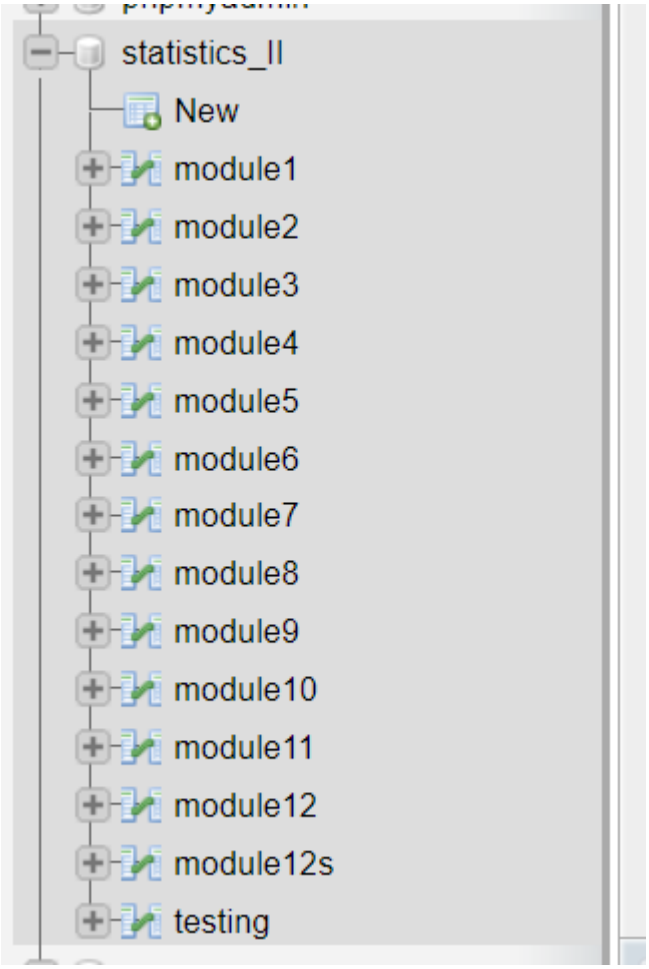


4. Navigate to the SQL tab

Enter the following code to create tables in the database. Change module1 to the correct module number (or any other name you would like to have for a table).

```
CREATE TABLE `module1` (
  `user_id` text NOT NULL,
  `event` text NOT NULL,
  `label` text NOT NULL,
  `correct` text NOT NULL,
  `question` text NOT NULL,
  `answer` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

5. Select Go at the bottom to run the code and repeat as needed for more tables.
6. The tables should now appear here



RMarkdown Code to Connect to the Database

You will need the following code in the setup chunk:

```
### FILL IN YOUR DATABASE DETAILS HERE ###

##############################################################################

pool <- dbPool(
  drv = RMySQL::MySQL(), ## Check that this corresponds to the type of database you are using (MySQL/MariaDB, etc.)
  dbname = "YOUR DATABASE NAME",
  host = "DATABASE IP",
  username = "USERNAME", ## Make sure the user has read & write rights for the selected database
  password = "PASSWORD"
)

dbtable = "THE TABLE YOU WANT TO STORE QUIZ/EXERCISE DATA IN FOR THIS MODULE"
##############################################################################

select_query = paste("SELECT * FROM", dbtable)
```

Ensure that you have the necessary packages to connect to the database: RMariaDB, DBI and pool.

To write data to the database when students interact with quizzes and coding exercises, put the following code below the connection code in the setup chunk:

```
## Recording data to database
new_recorder <- function(tutorial_id, tutorial_version, user_id, event, data) {
    cat(user_id, ", ", event, ",", data$label, ", ", data$answer, ", ", data$correct, "\n", sep = "", append = TRUE)

d_tibble <- tibble::tibble(
user_id  = user_id,
event = event,
label = data$label,
correct = data$correct,
question = data$question,
answer = data$answer
  )

## send to mysql
dbWriteTable(pool, "testing", d_tibble, append=TRUE, row.names = FALSE)}

options(tutorial.event_recorder = new_recorder)
```

And to read from the database, you can use this in each relevant code chunk: data <- dbGetQuery(pool, select_query). The data from the table chosen in the setup chunk will then be accessible in a data frame called data.

This data frame can be used to generate plots showing student answers for quizzes or progress with coding exercises.

```
```{r, Quiz2O, echo = FALSE}
plotOutput("Q2")
```

```{r, Quiz2R, context="server", echo = FALSE, warning = FALSE, message=FALSE, out.width="100%", fig.align = "center"}

## Write the correct answer here (it must be an exact match inside quotation marks)
correct_answer = "`lm(affect ~ poly(stress, 2) + performance)`"

output$Q2 <- renderPlot({

  data <- dbGetQuery(pool, select_query)

  answers <- subset(data, data$label == "Quiz2",) ## Make sure this is the label of the question
  answers[answers==""] <- NA
  answers <- na.omit(answers)

  answers_count <- as.data.frame(answers %>%
  count(answer))
  total_n = nrow(answers)
  answers_count$percentage <- (answers_count$n/total_n)*100
  answers_count$correct <- ifelse(answers_count$answer == correct_answer, "Correct", "Incorrect")

  ggplot(answers_count,
         aes(x = percentage,
             y = answer,
             fill=correct
             )
         ) +
    geom_col(width=0.6) +theme_minimal() + scale_fill_brewer(palette="Paired", direction=-1)  +
    xlab("Percentage (%)") + ylab("Answer") + labs(fill = "Correct")
})
```
```