Connecting to PostgreSQL Database with R using RPostgreSQL

R, a powerful statistical programming language, can seamlessly integrate with various databases to facilitate data analysis and manipulation. In this tutorial, we'll explore how to connect to a PostgreSQL database using the "RPostgreSQL" package, a robust R package that serves as a bridge between R and PostgreSQL databases.

Prerequisites

Before we begin, make sure you have the following:

1. **R Installed:** Download and install R from CRAN.
2. **RStudio (Optional):** For a more integrated R development environment, you can use RStudio, available for download here.

Installing and Loading RPostgreSQL

Our first step is to install and load the "RPostgreSQL" package. This package acts as a connector, enabling R to communicate with PostgreSQL databases. Open your R environment and execute the following commands:

```
# Install the RPostgreSQL package
install.packages("RPostgreSQL")

# Load the RPostgreSQL package
library(RPostgreSQL)
```

Connecting to PostgreSQL Database

Establishing a connection to the PostgreSQL database requires specific parameters, including the host, port, database name, username, and password. Let's set up the connection:

```
# Connection parameters
host <- "82.165.61.139"
port <- 5430
dbname <- "Veg"
user <- "TechBio"
password <- "T3chB10!"

# Create a connection
con <- dbConnect(dbDriver("PostgreSQL"),
                 host = host,
                 port = port,
                 dbname = dbname,
                 user = user,
                 password = password)
```

Now that we're connected, we'll perform basic database operations, including listing tables, inserting data, querying, and joining tables. Please note that you can copy and paste the above code to connect for both exercises.

Inserting Data into PostgreSQL Database

INSERT Data from a Dataframe

In real-world scenarios, you often need to insert data from an R dataframe into a PostgreSQL table. Here's how you can do it:

```
# Sample dataframe
new_data <- data.frame(
  project = c("Project1", "Project2", "Project3"),
)

# Insert dataframe into the table
dbWriteTable(
    con,
    name = "body.Project",
    value = new_data,
    row.names = FALSE,
    append = TRUE,
)

cat("Additional data inserted successfully.\n")
```

This step involves creating a new dataframe (`new_data`) and using the `dbWriteTable` function to add this data to the existing PostgreSQL table.

INSERT Data from a CSV/Excel Document

If your data is in a CSV or Excel document, you can use the `read.csv` or `readxl` packages to read the data and then insert it into the PostgreSQL table:

```
# Install and load necessary packages
install.packages("readxl")
library(readxl)

# Read data from CSV or Excel
csv_data <- read.csv("path/to/your/data.csv")
# OR
excel_data <- read_excel("path/to/your/data.xlsx")

# Insert data into the table
dbWriteTable(con, name = "your_table", value = csv_data, row.names = FALSE,
append = TRUE)

cat("CSV/Excel data inserted successfully.\n")
```

Reading tables from PostgreSQL Database

You can read the content of a single table or multiple tables from a PostgreSQL database using the `dbReadTable` function:

```
# Read a single table
single_table <- dbReadTable(con, "your_table")

# Read multiple tables
multiple_tables <- dbReadTable(con, c("table1", "table2"))
```

Please note, that this does not perform a `JOIN` operation. It simply reads the content of the table(s) and stores it in a dataframe. For this, please make use of the following sections.

Selecting Data from PostgreSQL Database

Retrieving specific data from a PostgreSQL table involves executing a `SELECT` query. For instance, selecting all records where age is greater than 25:

```
# Perform SELECT query
selected_data <- dbGetQuery(con, "SELECT * FROM your_table WHERE age > 25")

# View the selected result
print("Selected data:")
print(selected_data)
```

This step demonstrates how to filter and retrieve specific data from a PostgreSQL table using a `SELECT` query.

Joining Tables in PostgreSQL Database

Assume you have two tables (`table1` and `table2`) with a common column (`id`). You can perform an `INNER JOIN` using the "RPostgreSQL" package:

```
# Perform JOIN query
join_query <- "SELECT * FROM table1 INNER JOIN table2 ON table1.id =
table2.id"
joined_result <- dbGetQuery(con, join_query)

# View the joined result
print("Joined data:")
print(joined_result)
```

This step showcases how to combine information from two tables using an `INNER JOIN` based on a shared column (`id`).

Parsing `SELECT` or `JOIN` Query Result into a Dataframe

After executing a `SELECT` or `JOIN` query, you can parse the result into an R dataframe for further analysis:

```
# Perform SELECT query
select_query <- "SELECT * FROM your_table"
selected_data <- dbGetQuery(con, select_query)

# View the selected result
print("Selected data:")
print(selected_data)

# Parse the result into a dataframe
result_dataframe <- as.data.frame(selected_data)

# Display the dataframe
cat("Dataframe created from the selected result:\n")
print(result_dataframe)
```

In this step, we execute a `SELECT` query to retrieve data from the PostgreSQL table (`your_table`). The result is then printed to the console. Following that, we use the `as.data.frame` function to parse the query result into an R dataframe (`result_dataframe`). The dataframe is then printed to the console, making it easier to work with the data using R's data manipulation capabilities.