Importación y ejecución de consultas básicas de datos de vehículos eléctricos en MySQL

José Manuel Rodríguez Vélez

2024-11-02

Introducción

En este documento, describiré los pasos que seguí para importar un archivo CSV a una base de datos MySQL utilizando RStudio y Docker. El archivo CSV fue obtenido de Kaggle, y el proceso incluyó la creación de un proyecto en RStudio, la ejecución de un contenedor Docker con MySQL y la ejecución de un script en R para detectar la estructura de datos e insertar los datos en una tabla SQL.

Paso 1: Obtención del archivo CSV

Descargué el archivo CSV de Kaggle. Este dataset contenía información relacionada con coches eléctricos y sesiones de carga. Ver https://www.kaggle.com/datasets/valakhorasani/electric-vehicle-charging-patterns/data.

Paso 2: Configuración del proyecto en RStudio

Creé un nuevo proyecto en RStudio para organizar de forma eficiente mis scripts y trabajo. Esto me permitió mantener un flujo de trabajo estructurado para la importación y análisis de datos.

Paso 3: Ejecución de un contenedor MySQL en Docker

Para gestionar la base de datos, inicié un contenedor MySQL en Docker con el siguiente comando:

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=root -p 3306:3306 -d mysql:latest
```

Este comando configura una instancia de MySQL accesible en `localhost:3306` con la contraseña de root especificada.

Paso 4: Conexión de RStudio a MySQL e importación del CSV

Utilicé los paquetes de R **DBI** y **RMariaDB** para establecer la conexión a MySQL. El script de R que usé es el siguiente:

```r
# Instalar y cargar las librerías necesarias
if (!requireNamespace("DBI", quietly = TRUE)) install.packages("DBI")
if (!requireNamespace("RMariaDB", quietly = TRUE)) install.packages("RMariaDB")
```

```r
library(DBI)
library(RMariaDB)

# Conectar a la base de datos MySQL
con <- dbConnect(RMariaDB::MariaDB(),
                 host = "127.0.0.1",
                 port = 3306,
                 user = "root",
                 password = "root",
                 dbname = "EV_DS")

# Leer el archivo CSV
ruta_csv <- "/Users/marthinal/ev_ad_strategy/ev_charging_patterns.csv"
data_original <- read.csv(ruta_csv)

# Escribir los datos en la base de datos MySQL
dbWriteTable(con, "charging_data", data_original, overwrite = TRUE, row.names = FALSE)

# Leer los datos de la tabla de MySQL
data_mysql <- dbReadTable(con, "charging_data")

# Comparar los data.frames
comparacion <- all.equal(data_original, data_mysql)

if (isTRUE(comparacion)) {
  cat("Los datos se han insertado correctamente en la tabla de MySQL.\n")
} else {
  cat("Se encontraron diferencias entre los datos originales y los datos en MySQL:\n")
  print(comparacion)
}

# Desconectar de la base de datos
dbDisconnect(con)
```

Explicación del código

Este código en R permite importar datos desde un archivo CSV hacia una tabla en una base de datos MySQL. Además, verifica si los datos se han insertado correctamente comparando los datos originales del CSV con los almacenados en MySQL.

Pasos del Código

1. **Instalación y Carga de Librerías**: Se instalan y cargan las librerías **DBI** y **RMariaDB**, necesarias para la conexión y operaciones con MySQL desde R.
2. **Conexión a la Base de Datos MySQL**: Se establece una conexión con la base de datos MySQL, configurando el host, puerto, usuario, contraseña y nombre de la base de datos.
3. **Lectura del Archivo CSV**: El archivo CSV es leído y almacenado en un `data.frame` en R, que representa los datos que se importarán a MySQL.
4. **Inserción de Datos en MySQL**: Se crea una tabla en MySQL y se insertan los datos del CSV. Si la tabla ya existe, se reemplaza por completo con los nuevos datos.

5. **Lectura de Datos desde MySQL**: Después de la inserción, el código lee la tabla desde MySQL y la almacena en otro `data.frame` para realizar la comparación.

6. **Comparación de Datos**: Se comparan los `data.frames` del CSV original y el de MySQL para verificar que los datos se hayan insertado correctamente. Si hay diferencias, se informa detalladamente.

Este flujo garantiza que los datos del CSV se carguen correctamente en la base de datos y permite confirmar la consistencia de la inserción.

Verificación de la unicidad de User ID

Después de importar los datos, verifiqué si `User ID` se repetía en la tabla para evaluar si era necesario usar una clave primaria compuesta. Utilicé la siguiente consulta SQL:

```sql
SELECT `User.ID`, COUNT(*) AS num_ocurrencias
FROM charging_data
GROUP BY `User.ID`
HAVING num_ocurrencias > 1;
```

Resultados de la consulta

La consulta no devolvió resultados, lo que confirma que no hay `User ID` duplicados en la tabla `charging_data`. Esto significa que `User ID` es único en los registros y puede utilizarse como clave primaria sin necesidad de agregar otro campo. Si se hubieran encontrado duplicados, habría sido necesario considerar el uso de `Charging Start Time` junto con `User ID` para crear una clave primaria compuesta que garantizara la unicidad de cada sesión de carga.

Paso 5: Añadir la clave primaria

Dado que se confirmó que `User ID` es único en la tabla, procedí a añadir una clave primaria a la columna `User ID` con el siguiente comando SQL:

```sql
ALTER TABLE charging_data ADD PRIMARY KEY (`User.ID`);
```

Este comando establece `User ID` como la clave primaria de la tabla `charging_data`, asegurando que cada registro sea único y cumpla con las restricciones de integridad de la base de datos.

Consultas para Inspeccionar los Datos

En esta sección, se presentan algunas consultas SQL que se ejecutaron para analizar los datos y obtener información útil.

Número de sesiones por momento del día para usuarios de trayectos largos

Esta consulta muestra el número de sesiones de carga por momento del día en distintas ciudades para usuarios de trayectos largos.

```sql
SELECT `Charging.Station.Location`, `Time.of.Day`, COUNT(*) AS num_sessions
FROM charging_data
WHERE `User.Type` = 'Long-Distance Traveler'
GROUP BY `Charging.Station.Location`, `Time.of.Day`
ORDER BY `Charging.Station.Location`, num_sessions DESC;
```

Charging.Station.Location	Time.of.Day	num_sessions
Chicago	Evening	20
Chicago	Night	19
Chicago	Morning	18
Chicago	Afternoon	17
Houston	Night	30
Houston	Evening	28
Houston	Morning	19
Houston	Afternoon	19
Los Angeles	Night	26
Los Angeles	Morning	26
Los Angeles	Afternoon	25
Los Angeles	Evening	24
New York	Afternoon	26
New York	Morning	21
New York	Evening	18
New York	Night	13
San Francisco	Evening	31
San Francisco	Morning	22
San Francisco	Night	20
San Francisco	Afternoon	15

Día de la semana con mayor tráfico de viajeros de larga distancia por ciudad

Esta consulta muestra el día de la semana con mayor número de sesiones de carga para usuarios de trayectos largos en cada ciudad.

```sql
SELECT`Charging.Station.Location`,`Day.of.Week`, COUNT(*) AS num_sessions FROM charging_data WHERE`User
```

Charging.Station.Location	Day.of.Week	num_sessions
Chicago	Saturday	15
Chicago	Friday	13
Chicago	Wednesday	12
Chicago	Thursday	11
Chicago	Tuesday	10
Chicago	Sunday	7
Chicago	Monday	6
Houston	Saturday	17
Houston	Friday	16
Houston	Tuesday	15
Houston	Wednesday	14
Houston	Monday	12
Houston	Sunday	11

Charging.Station.Location	Day.of.Week	num_sessions
Houston	Thursday	11
Los Angeles	Monday	21
Los Angeles	Tuesday	18
Los Angeles	Saturday	17
Los Angeles	Wednesday	15
Los Angeles	Sunday	15
Los Angeles	Thursday	10
Los Angeles	Friday	5
New York	Tuesday	15
New York	Friday	15
New York	Sunday	13
New York	Thursday	12
New York	Saturday	11
New York	Wednesday	8
New York	Monday	4
San Francisco	Friday	17
San Francisco	Saturday	17
San Francisco	Wednesday	16
San Francisco	Tuesday	13
San Francisco	Thursday	10
San Francisco	Sunday	9
San Francisco	Monday	6

Ciudades con vehículos más antiguos para usuarios de tipo "Commuter"

Esta consulta muestra las ciudades donde los usuarios de tipo "Commuter" utilizan vehículos con una mayor antigüedad promedio.

```
SELECT `Charging.Station.Location`, AVG(`Vehicle.Age..years.`) AS avg_vehicle_age FROM charging_data WH
```

Charging.Station.Location	avg_vehicle_age
New York	3.87
Los Angeles	3.79
Chicago	3.70
San Francisco	3.52
Houston	3.20

Modelos de vehículos ordenados por distancia promedio recorrida desde la última carga

Esta consulta muestra los modelos de vehículos ordenados por la distancia promedio recorrida desde la última carga, lo que ayuda a identificar qué vehículos tienden a recorrer más kilómetros entre sesiones de carga.

```
SELECT `Vehicle.Model`, AVG(`Distance.Driven..since.last.charge...km.`) AS avg_distance_driven FROM cha
```

Vehicle.Model	avg_distance_driven
Hyundai Kona	161.29
Tesla Model 3	156.70

Vehicle.Model	avg_distance_driven
Chevy Bolt	156.69
BMW i3	147.31
Nissan Leaf	145.81