DATA422-W8-82171165
Assignment Submission Report

David Ewing (82171165)

2024-09-16

Overview

This report outlines the deliverables required for the SQL assignment. The assignment involves interacting with a PostgreSQL database, retrieving data, visualising it, performing joins, and analysing query performance. The implementation uses an `.Renviron` file for secure database credentials and an R script to perform the various operations.

Deliverable 0: Database Connection

The database connection is established using the following environment variables stored in the `.Renviron` file:

```
PG_HOST = mathmads.canterbury.ac.nz
PG_PORT = 8909
PG_USR = student_data422
PG_PASS = readonly
```

The R script connects to the PostgreSQL database securely by referencing these environment variables.

```
> #  environment variables (CONFIRM in environment values)
> dbname <- Sys.getenv("PG_DBNAME", "Jeff")
> host <- Sys.getenv("PG_HOST", "mathmads.canterbury.ac.nz")
> port <- Sys.getenv("PG_PORT", "8909")
> user <- Sys.getenv("PG_USR", "student_data422")
> password <- Sys.getenv("PG_PASS")
> # connection object via environment variables (.Fenviron)
> con <- dbConnect(
+   Postgres(),
+   dbname = dbname,
+   host = host,
+   port = port,
+   user = user,
+   password = password
+ )
> # confirm connection status
> if (!dbIsValid(con)) {
+   stop("FAILED connection to PostgreSQL database.")
+ } else {
+   print("SUCCESSFUL connection to PostgreSQL database.")
+ }
[1] "SUCCESSFUL connection to PostgreSQL database."
```

Deliverable 1: Listing Tables

The script lists all tables in the connected database using the following code:

```
tables <- dbListTables(con)
print(tables)
```

Specific results:

```
#--------------------------------------------------------------------------------
> # DELIVERABLE 1: List all tables in the connected database
> tables <- dbListTables(con)
> print(tables)
 [1] "actor"                        "address"                      "category"
 [5] "country"                      "customer"                     "film"
 [9] "film_category"                "inventory"                    "language"
[13] "rental"                       "staff"                        "store"
[17] "customer_list"                "film_list"                    "nicer_but_slower_film_list"
[21] "sales_by_store"               "staff_list"
```

Deliverable 2: Listing Fields in a Table

The script lists all fields from the `rental` table using the following code:

```
fields <- dbListFields(con, "rental")
print(fields)
```

Specific results:

```
> #--------------------------------------------------------------------------------
> # DELIVERABLE 2: List all the fields in a table
> fields <- dbListFields(con, "rental")
> print(fields)
[1] "rental_id"    "rental_date"  "inventory_id" "customer_id"  "return_date"  "staff_id"
```

Deliverable 3: Pulling Data

The script pulls data from the `rental` table using the following SQL query:

```
SELECT rental_id, rental_date, inventory_id, customer_id FROM rental LIMIT 10;
```

This retrieves the first 10 rows of the specified fields.
Specific results:

```
> #----------------------------------------------------------------------------
> # DELIVERABLE 3: Pull some data from the 'rental' table as an example
> # SELECT:      SQL to 'pull some data'
> # rental_id:   Unique ID.
> # rental_date: The date of  rental occurred.
> #inventory_id: ID of the rented item.
> #customer_id:  ID of the customer who rented the item.
> #LIMIT 10:     clause to limit data
> query <- "SELECT rental_id, rental_date, inventory_id, customer_id FROM rental LIMIT 10"
> data <- dbGetQuery(con, query)
> print(data)
   rental_id         rental_date inventory_id customer_id
1          2 2005-05-24 22:54:33         1525         459
2          3 2005-05-24 23:03:39         1711         408
3          4 2005-05-24 23:04:41         2452         333
4          5 2005-05-24 23:05:21         2079         222
5          6 2005-05-24 23:08:07         2792         549
6          7 2005-05-24 23:11:53         3995         269
7          8 2005-05-24 23:31:46         2346         239
8          9 2005-05-25 00:00:40         2580         126
9         10 2005-05-25 00:02:21         1824         399
10        11 2005-05-25 00:09:02         4443         142
```

Deliverable 4: Data Visualisation

Using `ggplot2`, a bar chart is generated to visualise the number of rentals per customer. The following plot is created:

```
ggplot(data, aes(x = factor(customer_id))) +
  geom_bar() +
  xlab("Customer ID") +
  ylab("Number of Rentals") +
  ggtitle("Number of Rentals per Customer") +
  theme_minimal()
```

Specific Results:

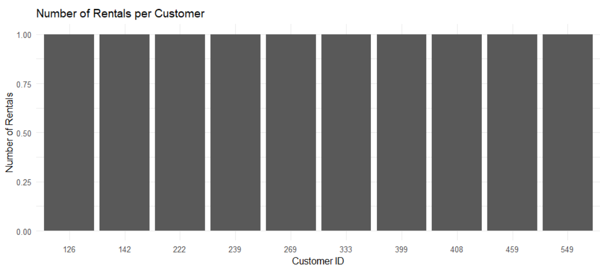

Figure 1: ggplot

Deliverable 5: Join Operation

A SQL `JOIN` (default) is performed between the `rental` and `customer` tables, retrieving the first and last names of customers who rented items. The following SQL query is used:

```
SELECT rental.rental_id, rental.rental_date, customer.first_name, customer.last_name
FROM rental
JOIN customer ON rental.customer_id = customer.customer_id
LIMIT 10;
```

Specific results:

```
> #---------------------------------------------------------------------------
> # DELIVERABLE 5: Perform a JOIN between 'rental' and 'customer' tables to get customer nam
> # INNER JOIN (SQL default):
> # JOIN customer ON rental.customer_id = customer.customer_id
> join_query <- "
+   SELECT rental.rental_id, rental.rental_date, customer.first_name, customer.last_name
+   FROM rental
+   JOIN customer ON rental.customer_id = customer.customer_id
+   LIMIT 10
+ "
> joined_data <- dbGetQuery(con, join_query)
> print(joined_data)
   rental_id         rental_date first_name   last_name
1          2 2005-05-24 22:54:33      Tommy     Collazo
2          3 2005-05-24 23:03:39     Manuel     Murrell
3          4 2005-05-24 23:04:41     Andrew       Purdy
4          5 2005-05-24 23:05:21    Delores      Hansen
5          6 2005-05-24 23:08:07     Nelson Christenson
6          7 2005-05-24 23:11:53  Cassandra     Walters
7          8 2005-05-24 23:31:46     Minnie      Romero
8          9 2005-05-25 00:00:40      Ellen     Simpson
9         10 2005-05-25 00:02:21      Danny        Isom
10        11 2005-05-25 00:09:02      April       Burns
```

Deliverable 6: Execution Plan Analysis

An `EXPLAIN` statement is executed to analyse the performance of the `JOIN` query. The query plan is printed as follows:

```
EXPLAIN SELECT rental.rental_id, rental.rental_date, customer.first_name, customer.last_name
FROM rental
JOIN customer ON rental.customer_id = customer.customer_id;
```

Specific Results:

```
> #--------------------------------------------------------------------------------
> # Investigate the execution plan for the JOIN query
> # results are in dataframe
> explain_query <- "
+   EXPLAIN SELECT rental.rental_id, rental.rental_date, customer.first_name, customer.last_
+   FROM rental
+   JOIN customer ON rental.customer_id = customer.customer_id
+ "
> execution_plan <- dbGetQuery(con, explain_query)
> print(execution_plan)
                                                              QUERY PLAN
1                    Hash Join  (cost=22.48..375.33 rows=16044 width=25)
2                 Hash Cond: (rental.customer_id = customer.customer_id)
3         ->  Seq Scan on rental  (cost=0.00..310.44 rows=16044 width=14)
4                        ->  Hash  (cost=14.99..14.99 rows=599 width=17)
5          ->  Seq Scan on customer  (cost=0.00..14.99 rows=599 width=17)
```

Conclusion

The `.Renviron` file and `00_initialisation.R` script together fulfil all the deliverables for this assignment, including secure database access, querying data, visualising results, and analysing query performance.