SQL and Dataframes with Julia

Tony Fraser : 10Sept2023

Assignment Overview

Utilize Visual Studio Code, Quarto markdown, SQL, and dataframes to navigate Julia fundamentals.

Conclusions

Amidst machine learning giants like TensorFlow and spark.ml, Julia distinguishes itself for tailored model designs. It's sleek, rapid, and bolstered by industry leaders like NVidia for GPU operations.

Julia Insights

1. **Positioned between Giants:** Julia sits between R and Python. Its syntax hints at Python, but the coding experience is reminiscent of R.
2. ***Blazing Speed:** Julia applications compile into fast and light C binaries. Their performance amplifies even further when tailored for specific GPUs.
3. **GTK Synergy:** In collaboration with GTK, Julia exhibits potential in software innovation — it could reshape tools like GIMP or Photoshop.
4. **Specialized Focus:** Julia excels in managing series and dataframes. Yet, areas like logging, markdown chunk processing, and comprehensive LaTeX support are secondary to its HPC features.
5. **Adoption Barriers:** Julia faces the same adoption challenges as Python and R. It is not object-oriented, presents a steep learning curve, and its integration into CI/CD pipelines can be intricate. It doesn't quite align with enterprise-friendly languages such as Object-Oriented Python, Scala, or C#.

Load Data

Hosted on Kaggle, this is the European Castle data set. And after this code block runs, data will be in both dataframes, and in the sqlite database.

Table headers:

- **CastleDetails**: Name, Latitude, Longitude, Adminstrative_Area_Level_1, Adminstrative_Area_Level_2, Municipality, Country, Postal_Code, Address, Editorial_Summary, Open_Hours, User_Rating, Wheelchair_Accessible_Entrance, Phone_Number, International_Phone_Number, Website
- **CastleUserReviews**:Name, Author, Rating, Review

```
packages = ["CSV", "DataFrames", "SQLite", "HTTP", "Gadfly", "IJulia"]
for pkg in packages Pkg.add(pkg)  end
using CSV, DataFrames, SQLite, HTTP, Gadfly

b = "https://raw.githubusercontent.com/tonythor/cuny-datascience/develop/data/eurocastles"
castle_details_content  = HTTP.get("$(b)/castle_details.csv").body
castle_user_reviews_content  = HTTP.get("$(b)/castle_user_reviews.csv").body

# castle_user_reviews_content <- comes back as a byte array of type ::Vector{UInt8}, to
# convert to a string do this: `content_string = String(castle_user_reviews_content)`

# Read the content into DataFrames
castles_df = CSV.File(IOBuffer(castle_details_content))      |> DataFrame
reviews_df = CSV.File(IOBuffer(castle_user_reviews_content)) |> DataFrame

# Initalize SQLite, an in memory sql database.
db = SQLite.DB("castles.db")

# Create tables and insert data from DataFrames into SQLite database
SQLite.load!(castles_df, db, "CastleDetails")
SQLite.load!(reviews_df, db, "CastleUserReviews")

#Create a little wrapper function that queries SQL and always returns a dataframe.
function q(query::String, db::SQLite.DB)
    result = DBInterface.execute(db, query)
    # |> DataFrame
    return result
end
```

```
    Resolving package versions...
  No Changes to `~/.julia/environments/v1.9/Project.toml`
  No Changes to `~/.julia/environments/v1.9/Manifest.toml`
    Resolving package versions...
  No Changes to `~/.julia/environments/v1.9/Project.toml`
  No Changes to `~/.julia/environments/v1.9/Manifest.toml`
    Resolving package versions...
  No Changes to `~/.julia/environments/v1.9/Project.toml`
  No Changes to `~/.julia/environments/v1.9/Manifest.toml`
    Resolving package versions...
  No Changes to `~/.julia/environments/v1.9/Project.toml`
  No Changes to `~/.julia/environments/v1.9/Manifest.toml`
    Resolving package versions...
  No Changes to `~/.julia/environments/v1.9/Project.toml`
  No Changes to `~/.julia/environments/v1.9/Manifest.toml`
    Resolving package versions...
  No Changes to `~/.julia/environments/v1.9/Project.toml`
  No Changes to `~/.julia/environments/v1.9/Manifest.toml`
```

```
q (generic function with 1 method)
```

```
castles_query = """
  select Name, Municipality, Country, User_Rating
  from CastleDetails
  order by User_Rating, Country, Name
"""

ratings_query = """
  select Name, Rating
  from CastleUserReviews
  where Rating is not null
"""

# c_r_join = """
# SELECT
#     cd.Country,
#     cd.Municipality,
#     COUNT(cd.Name) AS TotalCastles,
#     AVG(cd.User_Rating) as AvgUserRating
# FROM
#     CastleDetails cd, CastleUserReviews cr
# INNER JOIN
#     CastleUserReviews cr ON cd.Name = cr.Name
# GROUP BY
#     cd.Country, cd.Municipality
# ORDER BY
#     AvgUserRating, cd.Country, cd.Municipality
# """

cas = q(castles_query , db)
rat = q(ratings_query, db)
# c_r_join_group = execute_query(c_r_join, db)
```

```
# using Gadfly, DataFrames
# ratings_only = """
#   select Rating
#   from CastleUserReviews
#   where Rating is not null
# """
# rat_o = execute_query(ratings_only, db)
# p = Gadfly.plot(rat_o, y=:Rating, Geom.boxplot)
# display(p)
```