Sara Cruz – December 2024

Atlanta United FC - Data Engineering Scenario Project

This file is a guide for the zip file and my approach to the task.
Thank you for the chance to work on this project!

Sara Cruz – December 2024

Project Information

Zip File:

- *Folders:*
  - *backend (Python API) (localhost:8000/query)*
  - *frontend (React App) (localhost:3000/home)*
    - *./src/Components (includes the .jsx files for each page)*
  - *dev_test_app (extracted csv files)*
  - *SQL Sever (Placeholder in case I needed a Dockerfile) (Used the SQL Server Image)*
  - *Utilities (User made library to communicate with SQL Server)*
- *Files:*
  - *Docker-compose (Main File to Run)*
  - *Dockerfile (for extract_atl_data.py)*
  - *extract_atl_data.py (main script for data pipelines)*
  - *./frontend/src/app.js (main file for visualization)*
  - *SaraCruz_AU_DE.docx*

Database Location:

- *Host: LocalHost*
- *Username: 'sa'*
- *Password: 'tEST1234'*
- *Server: 'sql_server'*

Python Script: 'final_atl_de_proj/extract_atl_data.py'

- *Structure of Script:*
  - *Library Imports*
    - *Utilities Library is user created and I have used it for all data upload/extraction to SQL Server.*
    - *SQL Server Connection: connection = sql_server()*
    - *Create database: connection.create_datebase()*
    - *Upload Data: connection.sql_query_bt()*
      - *It first builds queries associated to creating, and merging tables.*
      - *Then creates the staging table + production table through the SQL Server connection.*
      - *The staging table is first created. Then the merging statement is used for the final upload to the production table.*
      - *This function takes into account any merging or updates to records. (If any errors occur during merging and error is displayed).*

Sara Cruz – December 2024

Project Overview

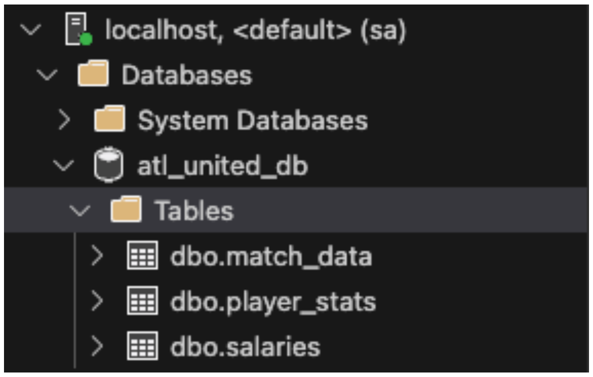
To be able to complete the project in a timely manner, I downloaded sample data from the provided website (*https://app.americansocceranalysis.com/#!/mls*) to initiate my pipelines.

If I had more time, I would investigate the website a bit more and decide on one of two avenues:

- *Developing a web-scrapping tool to get the data*
- *Dig deeper into the GitHub repo to find the endpoints where the data originates from.*

The data pipelines can be found in: extract_atl_data.py

- **Data Ingestion**:
  - This file showcases how data can be uploaded into a SQL Server instance.
  - For this specific example:
    - The databases are created first.
      - 'atl_united_db' : production data
      - 'atl_united_db_staging' : staging data.
    - Then the CSV files are extracted, table names, and merging columns are defined according to the csv file names.
      - Production Table: SELECT * FROM atl_united_db.dbo.match_data
        CSV File: 'american_soccer_analysis_mls_xgoals_games_2024-12-06.csv'
      - Production Table: SELECT * FROM atl_united_db.dbo.salaries
        CSV File: 'american_soccer_analysis_mls_salaries_players_2024-12-06.csv'
      - Production Table: SELECT * FROM atl_united_db.dbo.player_stats
        CSV File: 'american_soccer_analysis_mls_goals-added_players_2024-12-06.csv'



    - Lastly, the data is uploaded to the database. If the table defined by the user is not in the database the production + staging tables will be created, then the data is uploaded to the staging and finally merged to the production table *(additional information on the util.py library is provided in the next section).*

- *If I had more time:*
  - *I need to double check all the data types for each column.*
  - *Change the column names to more descriptive headers.*
  - *Build a system to establish id columns for all the tables.*

- **Pipeline Automation**:
  - There are two parts to the automation of this pipeline.
    - Data extraction + transformation.
    - Job Scheduling
      - This can be done through a VM. I have experience using both a Windows VM + TaskScheduler and a Linux VM + the cron.
  - The utils module (which I have built up through my work assignments) can be used to upload any data to user specified tables within the connected SQL Server database.

- **Data Visualization**:
  - For the data visualization, my approach was to build a simple interface where any front office employee can dive into the platform and get relevant data for their use case.
    - In the team page I envision a coach would want information regarding the season stats an individual player has.
      - Additionally, extra pages can be built to showcase previous game performance, player trends, player health + training information, etc.
    - In the league page, there is statistical information for teams + players around the league. I imagine two different use cases.
      - This can be the basis for game day reports to give to coaches, players, etc.
      - This can be the basis for scouting reports to aid in team transactions.

  - *If I had more time:*
    - *There are a few items that are not fleshed out:*
      - *League Page:*
        - *Page Styling/ Page Filtering*
      - *Team Page:*
        - *Staff Information*
      - *Additional Pages/Views*
        - *I have skeleton code for a scrolling horizontal footer where I would implement either recent MLS results or previous ATL match results.*
    - *I wasn't able to fully showcase the api endpoint I made that connects to the SQL Server database.*
      - *This endpoint is able to query any table in the db by defining specific table parameters in the request header.*

- *Once this connection is working, I would like to run latency tests on the entire app.*
- *Most of the data for the visualization is JSON data within the .jsx files and not from the tables extracted from the given website.*