Snowflake Data Loading from Chess Streaming Data

Description:

1) This project aims to extract data from chess.com APIs with a python script. The data is then sent to a Data Lake in the form of three json files. As soon as the data is loaded into the Data Lake, it is ingested into an SQS Queue and then sent to a Snowflake Data Warehouse via a Snowpipe Pipeline. Finally, the data is analysed with SQL queries in the Snowflake tables. The execution of the script is orchestrated by Airflow to operate every hour.
2) The approximately 600kb database is made up of streamers verified by chess.com, many of whom represent professional chess players. There are three main categories of games:
    a. blitz games, duration between 3 and 5min (not counting additional seconds)
    b. bullet games, duration of less than 2min
    c. rapid games, duration of more than 10min
3) Even if the database is very small, however, it allowed me to learn more about how data pipelines work. This size is partly explained by the fact that I use the free trial version of AWS S3.

Purpose:

Building an end-to-end Data pipeline to collect useful information about streamers on chess.com, storing this data in S3 and modelling it on Snowflake.

How to setup:

Snowflake project and S3 Data Lake project

The operation of the Data Lake and the Data Warehouse are explained in their corresponding file.

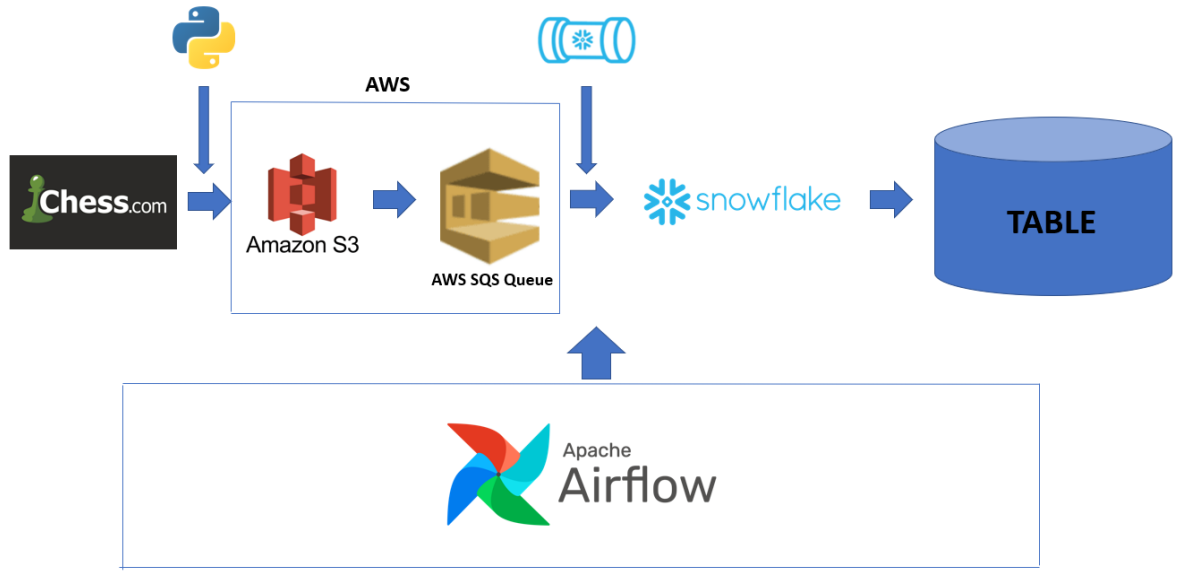
Python script

The python script first sends a request to the "streamers" API to get a list of all streamers on the platform. Then for each streamer the script sends a request for its general information and the statistics of these parts.

Airflow project

The Airflow python script allows you to launch the API python script every hour in order to update the list of streamers who are currently live as well as the statistics of their games. Due to the fact that Data Lake data is automatically sent to the Data Warehouse, only the Airflow scheduler role is used.

Architecture:



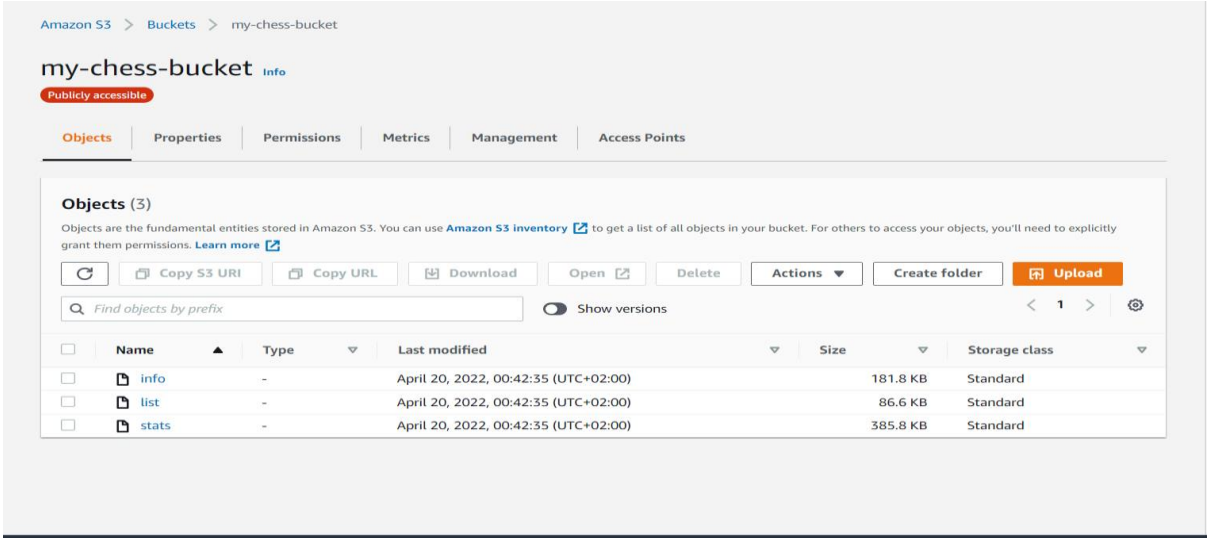
Step by step Implementation:

1. Snowflake project and S3 Data Lake project

- The goal is to load data from the chess.com APIs and store them into three different and raw json files. The data will then be transferred and transformed in a data warehouse in Snowflake.
- In order to not exceed the limit authorized by the free version of AWS S3, I focused on a small database of less than one gigabyte.
- The data lake receives data from a python script and stores it as is. To not lose information for other users of the database I decided not to delete information from json files for the moment.
- To send the data to Snowflake I created an SQS event notification which ingests the data into SQS Queue and which will then be processed by Snowflake.

Screenshot

Fig1. Screenshot of the bucket in AWS S3



Metadata

List file

```
{

  "username": "string",

  "avatar": "URL",

  "twitch_url": "Twitch.tv URL",

  "url":"member url's"

  "is_live": true

}
```

Info file

```
{

  "@id": "URL", // the location of this profile (always self-referencing)

  "url": "URL", // the chess.com user's profile page (the username is displayed with the original letter case)

  "username": "string", // the username of this player

  "player_id": 41, // the non-changing Chess.com ID of this player
```

```
  "title": "string", // (optional) abbreviation of chess title, if any

  "status": "string", // account status: closed, closed:fair_play_violations,
basic, premium, mod, staff

  "name": "string", // (optional) the personal first and last name

  "avatar": "URL", // (optional) URL of a 200x200 image

  "location": "string", // (optional) the city or location

  "country": "URL", // API location of this player's country's profile

  "joined": 1178556600, // timestamp of registration on Chess.com

  "last_online": 1500661803, // timestamp of the most recent login

  "followers": 17 // the number of players tracking this player's activity

  "is_streamer": "boolean", //if the member is a Chess.com streamer

  "twitch_url": "Twitch.tv URL",

  "fide": "integer" // FIDE rating

}
```

Stats file

```
{
  "chess_daily": {
    /* stats object for games of rules "chess" and "daily" time-class */
  },
  "chess960_daily": {
    /* stats object for games of rules "chess960" and "daily" time-class */
  },
  "chess_blitz": {
    /* stats object for games of rules "chess" and "blitz" time-class */
  },
  "tactics": {
      "highest": {
          "rating": "integer",
          "date": "timestamp"
      },
      "lowest": {
          "rating": "integer",
          "date": "timestamp"
      }
  },
  "lessons":{
      "highest": {
          "rating": "integer",
          "date": "timestamp"
      },
      "lowest": {
          "rating": "integer",
          "date": "timestamp"
      }
```

```
  },
  "puzzle_rush": {
      "daily":{
          "total_attempts": "integer",
          "score": "integer"
       },
      "best": {
          "total_attempts": "integer",
          "score": "integer"
       }
   }
}
```

Implementing Data Warehouse on Snowflake

The goal of this project is to route the data from the SQS Queue through a Snowflake Pipeline (Snowpipe), to transform this data into a relational data table and then to extract useful information through SQL queries.

Files

`create_table.sql` create the three different tables for the data
`snowpipe_S3_to_Snowflake` create the data pipeline and the stages to transfer the data from the SQS Queue
`sql_queries` transform in order to retrieve useful information from the data

General information

In this project we want to extract the three JSON objects "list", "stats" and "info" located in the S3 data lake "chess-bucket". To facilitate this transfer we have created an event trigger which allows the data to be stored in an SQS Queue which we must then use to transmit to the Data Warehouse. For this we have created a Snowpipe which copies data into a json table on Snowflake. From this json table we have created a table by recovering only the necessary columns which are:

list_table:

```
-username: varchar -- username of the streamer

-is_live: binary -- is the streamer currently on live
```

info_table:

```
-username: varchar  -- username of the streamer

-followers: numeric -- number of followers of twitch

-country: varchar  -- country of the streamer

-joined: date -- date the streamer joined chess.com
```

```
-location: varchar -- city of the streamer

-name: varchar -- full name of the streamer

-player_id: numeric

-status: varchar -- premium, admin, staff, basic

-title: varchar -- FIDE title

-primary_key: numeric
```

stats_table:

```
-last_blitz: numeric -- elo of the last blitz game (3 to 5min game)

-draw_blitz: numeric  -- number of draws in blitz games

-loss_blitz: numeric  -- number of defeats in blitz games

-win_blitz: numeric  -- number of wins in blitz games

-last_bullet: numeric  -- elo of the last bullet game (less than 2min game)

-draw_bullet: numeric  -- number of draws in bullet games

-loss_bullet: numeric  -- number of defeats in bullet games

-win_bullet: numeric  -- number of wins in bullet games

-last_rapid: numeric  -- elo of the last rapid game (over 10min game)

-draw_rapid: numeric  -- number of draws in rapid games

-loss_rapid: numeric  -- number of defeats in rapid games

-win_rapid: numeric  -- number of wins in rapid games

-FIDE: numeric -- official FIDE ranking

-primary_key: numeric
```

Results

Thanks to SQL queries we retrieved this information:

- Username of the best player by category (blitz, chess, bullet)
- Full name (or username if null) of the best player and his FIDE elo
- Average elo of premium, staff and basic players
- Number of professional players and their elo
- Average FIDE elo by their professional FIDE elo
- Best player currently on live