Big Data Analytics Final Project - CSCI 636 M02
Fall 2023
Final Project Report

Kevin Chow, Joseph Heaney, Rex Ocampo

Project Goal

The goal of this project is to ingest, analyze, and visualize match data from approximately 200,000 high ELO games from the popular online video game, League of Legends, in order to understand Win percentage by Side, First Objective Analysis, and Game State Side Correlation Comparison, using Amazon Web Services.

Which to choose: Data Lake or Data Warehouse?

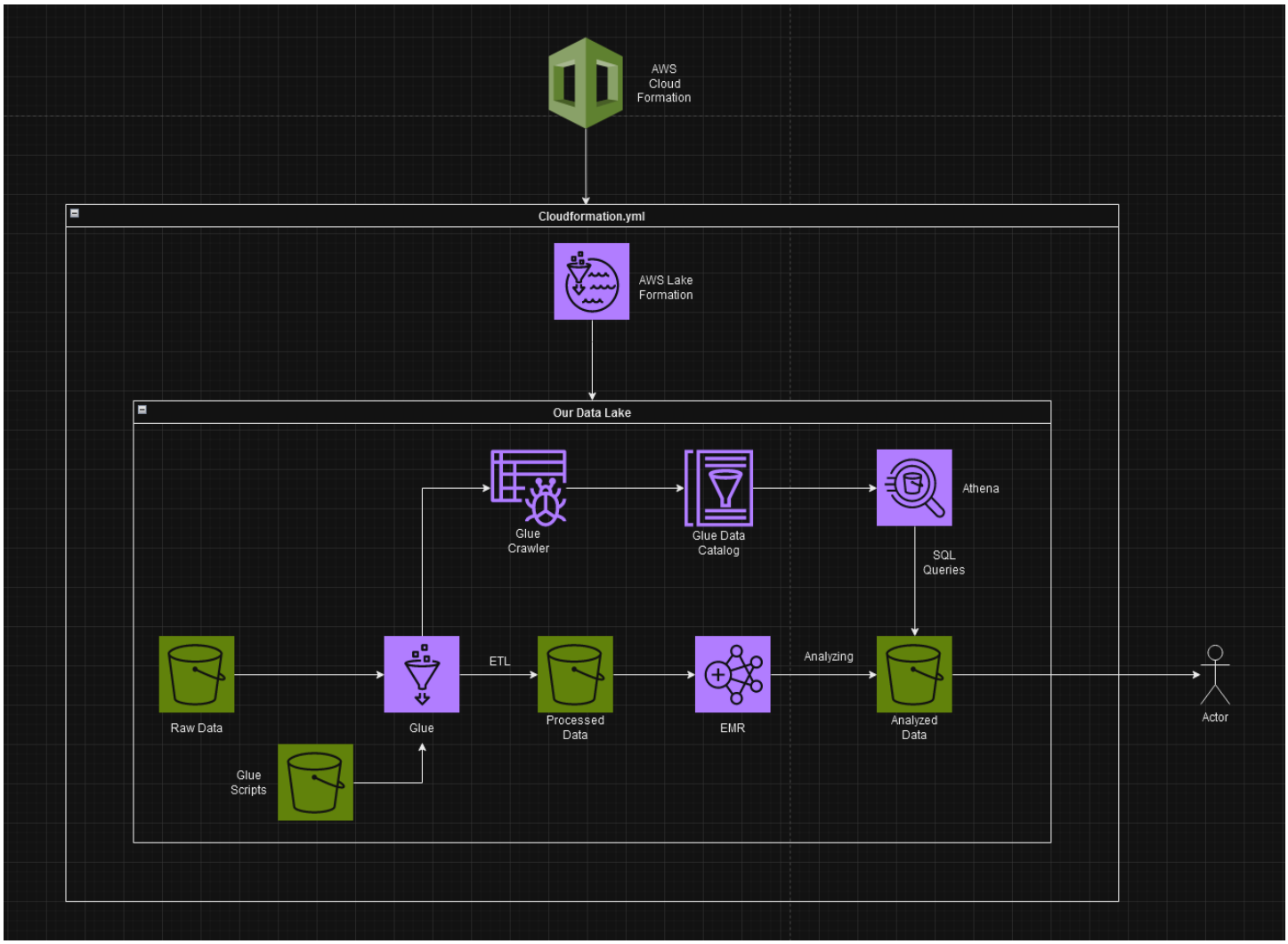
Point	Data Lake	Data Warehouse	Which to choose?
Data Type	Raw, Unstructure, Semistruture, Structured	Structured and Pre-process data	**Data Lake**
Schema	Schema-on-read, no predefined structure	Schema-on-write, predefined structure	**Data Lake**
Processing	On-demand processing	Batch processing	**Data Lake**
Data Agility	Accommodates diverse data formats without prior transformation	Requires data transformation before storage	**Data Lake**
Data Insight	Enables exploration of fresh ideas from raw and unprocessed data	Offers insight from processed, organized data	**Data Lake**

The Choice? Data Lake.

The choice is clear when going over the different points between choosing a data lake and data warehouse. Data lakes are suited for handling raw data, which is exactly what the dataset is. Another advantage is there is no schema within the dataset, so it is perfect for a data lake, and not a warehouse, which requires its data to have a schema predefined. Finally, the point is that on-demand data processing and data transformation that data lakes are capable of makes it the perfect choice for this particular dataset.

Data set was retrieved from this kaggle dataset
https://www.kaggle.com/datasets/gyejr95/league-of-legends-challenger-ranked-games2020

Architecture System Design Diagram



Data Lake Creation

In order to create a data lake, many AWS components will be used, such as Lake Formation, Glue, and S3. Creating a data lake manually is a labor intensive task, so a way to streamline the process would be to take advantage of AWS CloudFormation and Stacks.

CloudFormation is a service that allows for fast deployment of infrastructure in an automated manner through the use of templates. These templates describe the set of resources required for an infrastructure. The set of resources used is called a Stack, where all resources are managed as a single unit.

By uploading a template file called cloudformation.yml , a Stack is easily created and the Data lake formed.
The cloudformation.yml file used in this project can be found through these github links.

https://github.com/ramendude/Big-Data-Final-Project/blob/main/cloudformation.yml
https://github.com/kevinchow999/BigDataFinalProj/blob/main/cloudformation.yml
https://github.com/JoeHeaney/BigDataFinalProject/blob/main/cloudformation.yml

Connecting and Using AWS Services
From creating a Stack by using the provided cloudformation.yml, the Stack will take care of all data ingestion and analysis by taking in user-defined parameters and definitions for deployment, such as a KeyPair value for the EC2 instance and the necessary security groups for the EMR instances.
The Stack will then create and connect all the necessary services in order to complete its task, including:

S3 Buckets Creation and Setup:
The template created multiple S3 buckets in order to hold the requisite data:

'Finalproject-raw-data' holds the initial csv data.
'Finalproject-glue-script' holds all the scripts required for data processing.
'Finalproject-analytics' holds the parquet data generated from the glue jobs.
'Finalproject-results' holds the results generated from data analysis from the EMR cluster.
'Emr-log-bucket' holds log data from the EMR cluster.

EMR Cluster and EC2 Instance Creation and Setup:
The template created an EMR cluster that utilized m5.xlarge EC2 instances. One instance was created for both the master and the core group respectively. This cluster will be utilized for data processing and visualization through Spark and job submissions.

AWS Glue Database and Table Creation and Setup:
The template created an AWS Glue Database labeled 'my_database', where all the processed data from the glue jobs will reside.
A gluecrawler is then created in order to 'crawl' around in the database to infer the schema of the database and catalog the metadata into a data catalog through a table.

AWS Glue Jobs Creation and Submission:
The template created three glue jobs, labeled 'GlueJob' followed by a respective number and submitted in ascending order. These glue jobs use a specified script found in the S3 glue script bucket meant for their respective dataset.

These glue jobs take care of the process of reading the dataset, transforming, and loading into a chosen data format. When finished, the glue job creates a parquet file to hold the processed data and writes it to the S3 analytics bucket.

AWS Glue Script Breakdown

The glue script found in the 'Finalproject-glue-script' S3 bucket is responsible for data extraction, data transformation, and data loading functionality, or Glue ETL. The script will be submitted as a Glue Job and the breakdown of the script function is as follows:

Job Initialization:

The first portion of the script involves importing the necessary libraries and modules to create a Spark Session for a Glue Job.

Data Extraction:

The second portion of the script involves going through the S3 raw data bucket, and reading the specified dataset chosen. This data is then placed into a Pandas dataframe.

Data Transformation:

The third portion of the script involves transforming the data by ensuring that there is no duplicate match data. If there is a duplicate row, that row is then dropped.

Data Loading:

The fourth portion of the script involves writing the processed data into its respective parquet file and saving it onto the S3 analytics bucket.

Job Submission:

The final portion of the script is to submit the glue job through job.commit() so it can be run through AWS Glue.

Data Analysis using EMR through a Spark Job

Through the use of python script 'big_data_analytics.py', a Spark job can be submitted to the EMR cluster for data analysis. The breakdown of the script is as follows:

Job Initialization:

The first portion of the script involves the necessary libraries and modules to create a Spark Session for an EMR Step and the necessary tools for data analysis and visualization.

Data Reading, Assigning, and Merging:

The second portion of the script involves loading three S3 bucket paths for each of the respective parquet files for our data, and creating a respective Pandas dataframe for each parquet file. In order to not have the datasets mixed up, a new column will be assigned to each Pandas that corresponds to the 'Rank' that the dataset is from. Finally, all the datasets are merged into one big dataframe for data analysis.

Win Rate Analysis:
This section of code is focused around Win Rate between Blue Side and Red Side. The code takes the columns ‘blueWins’ and ‘redWins’ and finds the means for each. The result is then plotted onto a Pie chart using Matplotlib in order to visualize the Win Rate between the two sides across all three ranks.

Game Duration Analysis:
This section of the code is focused around the Game Duration. The code takes the ‘gameDuration’ column and finds the mean. The result is then converted into minutes. The code then takes the ‘gameDuration’ column and plots onto a histogram in order to see the normal distribution of game duration across all games. This analysis is key because it helps identify that the average game length takes about 24 minutes.

Blue Side vs Red Side Match Data Analysis:
This section of the code is focused around match data for each side. The code creates a list of key categories and finds the percentage for each side for that particular category. Each side data is then plotted onto a bar graph that compares the difference between the sides over 1%. Columns such as Wins, FirstTower, FirstBaron, FirstDragon, DragonKills, and BaronKills are plotted along the x axis and their percentages along the y axis.This analysis is crucial as it highlights the distinct priorities considered by each side and emphasizes specific match objectives that are crucial for securing a game victory.

Win Correlation Analysis of Structural Objectives:
We used the Pearson correlation coefficients for each variable in the dataset concerning the 'blueWins' outcome. This is achieved by utilizing the Pandas corr() function on the DataFrame, and the resulting correlation series is then sorted in descending order. The specific variables with correlation coefficients greater than 0.5 are isolated for further analysis. Subsequently, a new DataFrame is created using these selected variables, and the correlation matrix is computed using the Seaborn heatmap function. The resulting heatmap provides a visual representation of the strength and direction of the correlations, with lighter shades indicating stronger positive correlations. The analysis done was simple: destroy towers and inhibitors to have a greater chance at destroying the enemy’s base.

Win Correlation Analysis of Neutral Objectives and Game Events:
This segment of the analysis delves into the intricate correlation between various neutral objectives and critical in-game events. The code uses a dual-heat map presentation that compares the win correlations for the blue and red sides. The coding was similar to the last paragraph where Pearson method was used and implemented through the function corr() and heatmap() functions. Unlike the previous heatmap, we used more relevant variables such as FirstDragon, FirstHerald, FirstBaron, and FirstBlood. The resulting visualizations revealed that, as anticipated, both sides exhibit nearly identical yet inverted win correlations. Notably, the analysis reinforces the strategic significance of securing neutral objectives, such as dragon, herald, and baron, as they emerge as pivotal contributors to victory by enhancing team strength

through buffs and gold acquisition. This will directly impact the number of towerKills and inhibitorKills a team may achieve which was proven to be the most important factor for winning.

Submitting Queries through Athena:
Using AWS Glue, a database named 'finalproject_raw_data' is established, along with a corresponding table encompassing the complete schema of the project database. AWS Athena can then be employed for executing SQL queries on this database.

Conclusion

In conclusion, the analysis of ranked League of Legends games played in 2020 across the top three ranks reveals that there is no significant advantage for either the blue or red side. The examination encompassed various aspects such as win percentages, game duration, and correlations between in-game variables. Despite visual differences in some metrics, the overall trend suggests a balanced gameplay experience between both sides. The distinction between blue and red side advantages seems to be minimal, as both sides exhibit strengths and weaknesses that offset each other. It is crucial to note that this analysis is specific to the 2020 dataset and the top three ranks, and it may not extrapolate to other years or lower ranks where the game's meta and balance may vary. Additionally, it is important to consider potential changes to champions, items, and maps that can influence the dynamics of the game (meta) in subsequent years.