DSCI 351 Final Project

Aiden Bingham
12/02/2022

To provide some background context, I had started my project over 4 times as I couldn't find the right dataset with necessary values for mapreduce functionality I could apply. Whether there were too many columns or not enough integers provided, it took me some time before I could find the dataset I needed. My dataset on hatecrimes across the United States provided necessary information in string variables that I couldn't use functions with. Also, other Spotify datasets only provided rankings and streams which were too polarized to create averages or sum functions with. I had been increasingly overcomplicating my final project ideation by using ML modelling tools such as K-fold cross validation and cross-linear correlations for analysis that weren't needed. Finally, I concluded that this dataset would be best for distributing the necessary reduce functions I needed.

After partitioning the dataset into two tables with SQL, I cleaned and dropped extraneous values from both of them. Then, I created the display and explanation facility functions to appropriately show my terminology and process behind each visualization. These print functions were used with my sum and average reduce functions with Groupby statements for the final result. With all my graphs and charts created with SQL mappartitions and python reduce functions, I made a user main menu for people using my notebook to explore different categories. Whether it's beats per minute, duration in milliseconds, energy, danceability, loudness, liveness, valence, acousticness, speechiness, or popularity, the user can find the average, sum, or other analytics of these categories.

For this project, I initially used a Jupyter notebook but used Google colab for better formatting. I used a mix of Python and SQL with my python functions utilized for defining functions and user inputs while SQL selected the necessary elements for each data partition. I also started on a PHP dynamic page based on some experience working with HTML and front end web development, but realized that the page wasn't necessary for grading as my earlier project team had disbanded after a couple weeks after the midterm report. Since I started with multiple other datasets that ended up too big, I used many different applications such as MySQLworkbench and Node.js for creating my data partitions that ultimately weren't used in my final project. Obstacles and hurdles I faced ultimately helped me learn how to create this project as I went through many tutorials on HTML and dynamic processing to create my user main menu. Though I ended up receiving many errors that would result in my computer crashing while creating many tables and trying mapreduce functions in SQL, I finally used Colab

notebook to simultaneously use SQL and Python together to create my mapReduce functionality.

Even though I couldn't create my dynamic UI website that I intended, my explanation facilities are given before each function. My functions provide information such as top 50 popular songs, least 50 popular songs, loudest songs, quietest songs. With mapReduce functions, components such as duration or energy are given visualizations through graphs and charts by year or based on popularity. Each table has about 1000 rows of data from 1956 to 2019. After cleaning and dropping duplicate / extraneous values, I connected to the SQL database. Each SQL table was created from the partitions through for loops that inserted the values into each row.

To display my explanation facilities with the partitioned data, I created functions that give summaries with the final results. My mapreduce functions for average and sum used two mapPartition SQL lists and sorted them by key values. The average was appended into a new list that resulted in a final output. The mapReduce for average and sum used group by while divded by 2 to get the mean category. SQL statements are executed to get two lists that are mapreduced to make a new result. With both parts, a new graph is created through matplotlib. The mapreduce output to the user with this graph shown. Users can choose from the categories with menu items in Python to show bar graphs, scatter plots, and more. Top 50 artists with average popularity and sum popularity are different based on just the mapreduce functions in new lists to create the final output.

Similar to graphs, the charts are created with SQL statements that pull data from the data partitions and use mapreduce functions to create two lists. The results are appended into one list that are sorted by key in lambda function to create the top 10 in the list. The explanation facility shows how the final output was given by top 10, most loudest, or more charts. Least 10 popular songs have differences in code with year = ? to find specific year and appended final result is sorted by lambda. These are ranked by ascending or descending order. The user can see that it's higher and lower based on most or least popular songs. Sorted by popularity score in the explore by popularity function to find songs with a specific popularity score. These are similar functions for exploreby artists, song title, music genre, or year their music came out. In one program, users can utilize this function to choose which process to explore based on the Spotify top 2000's dataset. You can enter numbers to search by specific categories and visualizations that mapreduce functions are used on.