**Steps for implementing GitHub Aggregator**

**1. Importing Necessary Packages/Libraries:**

Import required libraries such as requests, pandas, pymysql, and datetime to set up the environment for data retrieval, processing, database interaction, and time handling.

**2. Taking User Input:**

Prompt the user to input the GitHub repository name, start date, and end date to customize the data retrieval process.

**3. Setting Default Values:**

Set default values for the start date ('2010-01-01') and end date (today's date) if the user does not provide them.

**4. Defining Variables:**

Initialize various variables such as access token, headers, run date, data frames, and dictionaries to store and manipulate data throughout the code execution.

**5. Fetching Commit Data from GitHub API:**

Construct URLs based on user input to send HTTP GET requests to the GitHub API and retrieve commit data within the specified date range.

**6. Pagination Handling:**

Iterate through multiple pages of commit data if the response from the API contains pagination, ensuring all relevant commits are retrieved.

**7. Processing Commit Data:**

Extract relevant information from the retrieved commit data, such as email IDs, company names, and dates, and store it in dictionaries for further analysis.

**8. Calculating Contribution Metrics:**

Calculate metrics like total contributions and unique contributors based on the processed commit data to gain insights into contribution patterns within the specified date range.

**9. Printing Results:**

Print various statistics and output, including total pages of commit data, total companies contributing, total contributions, unique contributors, languages used, and the output JSON data.

**10. Storing Data in Database:**

Establish a connection to a MySQL database and store the processed data for future retrieval and analysis. Insert the data into the database and commit the changes to ensure data integrity.

**11. Error Handling:**

Include a try-except block to handle exceptions gracefully and print informative error messages to aid in troubleshooting if any errors occur during execution.

**12. Execution Time:**

Calculate and print the execution time of the script to provide insights into its performance.