Group 9 - Iota
Assignment Requirements: Design Description
CS 4320
November 14th.

Goals:

The goal of this project is to implement several additional API endpoints to Augur. These will be integrated with the existing API and offer functionality not already included. While these new endpoints will not be associated with visualizations (out of scope), they will be implemented in a manner that can facilitate such developments at a later time. These new endpoints are as follows.

- Sustained contributors (proportion of non-new contributors)
    - How many existing contributors have in some way contributed to the project during a certain period of time. This will involve mapping a given contributor to all types of contributions, including commits, issues, and pull requests, and aggregating this data.
    - This is a time series and best suited to a standard metric.
    - Tables accessed in database query:
        - commits, issues, commit_comments, issue_comments, pull_requests, pull_requests_comments
- Release/tag frequency X
    - Average period of time between discrete tags or releases. This will involve comparing the times of various releases to each other and taking the average interval between them.
    - This is a time series and best suited to a standard metric.
    - Tables accessed in database query:
        - releases
- Proportion of merged/closed pull requests X
    - Ratio of merged pull requests over pull requests closed without merging. This will involve aggregating the total merged pull requests and closed pull requests per-repository.
    - This is a summary of the database and is likely best suited to a non-standard metric (route).
    - Tables accessed in database query:
        - pull_requests
- Sum of additions/removals per contributor per repository X
    - Net code additions for a particular contributor. This will involve taking the sum of the additions and the sum of deletions from a particular contributor and calculating the difference.
    - This information describes information about particular contributors and is best suited to a non-standard metric (route).
    - Tables accessed in database query:
        - contributors, commits

- Messages by users on issues/PRs
  - This is a general category. The following is information that is relevant to this concern.
  - All of these describe information about contributors in relation to messages and are likely best implemented as non-standard metrics (routes).
  - Messages in discussions by authors over time (absolute or proportional). This will involve aggregating the amounts of messages by each author for a certain time period.
    - Tables referenced
      - issues, pull_requests, message
  - Proportion of message counts on PRs that were merged vs closed. This will involve taking the sum of the messages on PRs that were merged and the sum of the messages on PRs that were closed and calculating the difference.
    - Tables referenced
      - pull_requests
  - Total of message lengths by author.
    - Tables referenced
      - pull_requests, issues, contributors

From an architectural perspective, all of these noted changes can be made in the form of database queries, as is done for the already present endpoints. Beyond this, it will be necessary both to provide a mechanism to specify parameters to the endpoints and provide a means for the result of the query to be returned back to the caller, as is necessary for the API extensions to be useful.

An example endpoint implementation can be found in api/metrics/pull_request.py the dev-ryan branch.

Tests:

Test cases will ensure that for the given input request, appropriate results are returned. Because no proposed API endpoints modify the database, it is not necessary to check for data integrity both before and after the test. Tests will ensure the correctness of the information returned and that the new endpoints access the database correctly.

Tests are located in their own folder in the github repository. See the tests directory on the dev-jafk5w branch.

Group 9 - Iota
Assignment Requirements: Design Description
CS 4320
November 14th.

SQL Query Examples for Endpoint Development:

While the purpose of this stage of development is not to entirely implement all details of the described endpoints, the following are some queries that could be used to retrieve the information required for some of their functionality.

- Proportion of merged/closed pull requests
    - This query returns a table of repos and the corresponding merged/closed pull request ratio for each repo. If there are no closed pull requests for the repo, then the ratio is null (division by 0)

```
SELECT
        a.repo_id, CAST(merged AS FLOAT) / NULLIF(CAST(closed AS FLOAT),0) AS
merged_to_closed_ratio FROM
        (SELECT
                repo_id, COUNT(pr_merged_at) AS merged
                FROM augur_data.pull_requests
                WHERE pr_merged_at IS NOT NULL
                GROUP BY repo_id) a
        INNER JOIN
        (SELECT
                repo_id, COUNT(pr_closed_at) AS closed
                FROM augur_data.pull_requests
                WHERE pr_merged_at IS NULL
                GROUP BY repo_id) b
        ON a.repo_id = b.repo_id
        WHERE a.repo_id = :repo_id;
```

Note: It may be prudent to select a specific row from this resultign table for use in augur; this is a more general query that can be easily selected from.

Group 9 - Iota
Assignment Requirements: Design Description
CS 4320
November 14th.

- Sum of additions/removals per contributor per repository.
    - This query will return a table of contributor email addresses mapped to the net contributions by that contributor. The table will be empty if there are no common contributors between the contributors and commits tables.

```
SELECT
        cntrb_email,SUM(cmt_added)-SUM(cmt_removed) AS net_contributions FROM
        augur_data.contributors,augur_data.commits
                WHERE cntrb_email=cmt_committer_email
                GROUP BY cntrb_email;
```

Note: It will be necessary to select a particular row from the result to get the results for a particular contributor of a given repository, which may be required depending on how the endpoint is implemented in augur.

- Release/tag frequency
    - Note that this query is currently hard-coded to return the number of releases per year for each repo. It should be improved upon in the future to support multiple time frames.

```
SELECT
        repo_id,
        COUNT(release_published_at) /
                ((EXTRACT (EPOCH FROM (:end_date - :start_date)))/60/60/24/365)
                AS releases_over_time
        FROM augur_data.releases
        WHERE release_published_at BETWEEN :start_date AND :end_date
        AND repo_id = :repo_id
        GROUP BY repo_id;
```