Name: Qiongwen(Dolly) Liu
Email: qiongwen7551@berkeley.edu
Education: Junior student at University of California, Berkeley
Language: English
LinkedIn: https://www.linkedin.com/in/qiongwen-liu-b44269223/

Program applied for:
PostgreSQL, Develop Performance Farm Benchmarks and Website (2022)

Abstract:

PostgreSQL already has up and running performance farm benchmarks which are fully functional. But it can benefit more from adding additional features to make it more usable.

1. Adding email notification/alert:
When we are testing the performance from a new code change of PostgreSQL, we can develop three level of notifications to the dev team: Red (the code contains bug that will prevent system from working correctly), Yellow (the code is good but the performance does not beat our benchmark, acceptable), Green (the code is good and it beat out benchmark, acceptable). For sending email to the dev team or our co-works, we can use a template to do so, by replacing the attribute within it, we can give it different levels we want.

2. Visualize performance data for the team:
A chart or graph can be better than a hundred words, data visualization is important for profiling and reporting any potential bottlenecks to a system. It gives people more sense of how the system performs and where we can do better. We can use Apache Superset for data visualization or use Kibana for it depending on our need (or some other tools).

3. Setup pressure test environments
Pressure testing is quite important for a system especially for a database. Usually we cannot do pressure tests in a working environment, so having a separate environment in our performance farm benchmarks will be beneficial. In which we can create a destructive testing benchmark and test every new change and compare it to it.

Proposal Timeline:

This week-by-week timeline provides a rough guideline of how the project will be done.

May 23 - June 13(Before coding starts):
Practice coding skills, especially for python with Django, learning about the structure of PostgreSQL system, and data visualization tools.

June 13 - June 19(Official coding period starts):
Familiarize yourself with the code and the community, the version control tool, the documentation and test system used.

June 20 - June 26:
Working on the email notification feature, integrate it with existing performance benchmarks with different levels.

June 27 - July 3:
Testing the email notification feature and making sure this new feature is deliverable.

July 4 - July 10:
Working on the data visualization feature, use Superset/Kibana to integrate with our testing data, implement different charts and graphics for different testing scenarios.

July 11 - July 17:
Testing on the data visualization feature, fix bugs if needed and make sure the feature quality is good.

July 18 - July 24:
Working on the pressure test environment setup, creating pressure test data and applying it to the PostgreSQL system, generating benchmarks for the existing system.

July 25 - July 29:
Targeting new code changes, apply them to the pressure test environment as well, compare the performance with the new pressure test benchmark and make conclusions based on that.

(Deadline July 29, Submission start from July 25)