TensorFlow Github Repository Activities

BY Kailei Lin

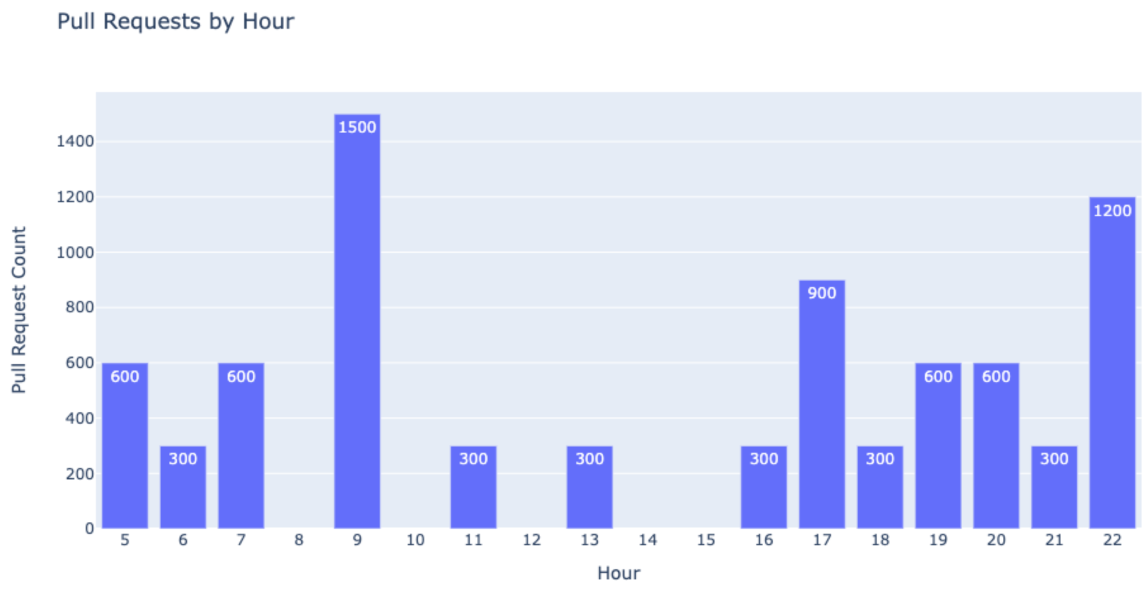
KPIs

- The average number of pull requests per contributor?
- The maximum number of commits per week?
- The minimum number of days of an issue being open?
- The maximum time between the creation and the merge of the pull requests?

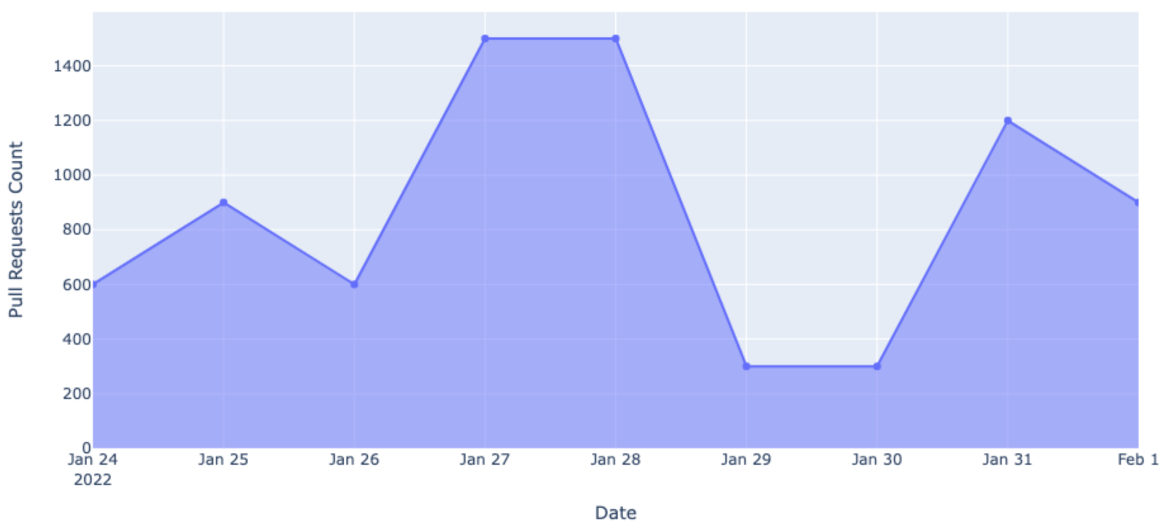
Activity

Pull Requests

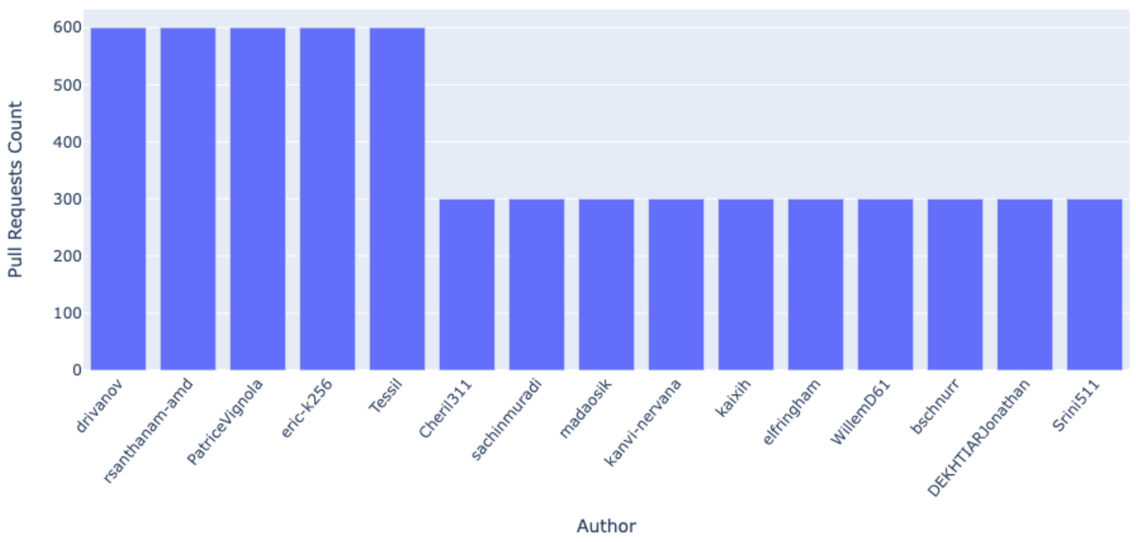
During the period of Januarary 24th 2022 to Feburary 2nd 2022, there are 26 unique requests. 9 am, 5 pm and 10 pm has the largest amount of requests. Between Date of Jan 27 to Jan 28, there are more than 1400 requests that have been made. User *drivanov, rsanthanam-amd, PatriceVignola, eric-k256, Tessil* are the top contributor to pull requests, which above 500 counts per person. The most used language for Tensorflow repository is C++.



Pull Requests by Date



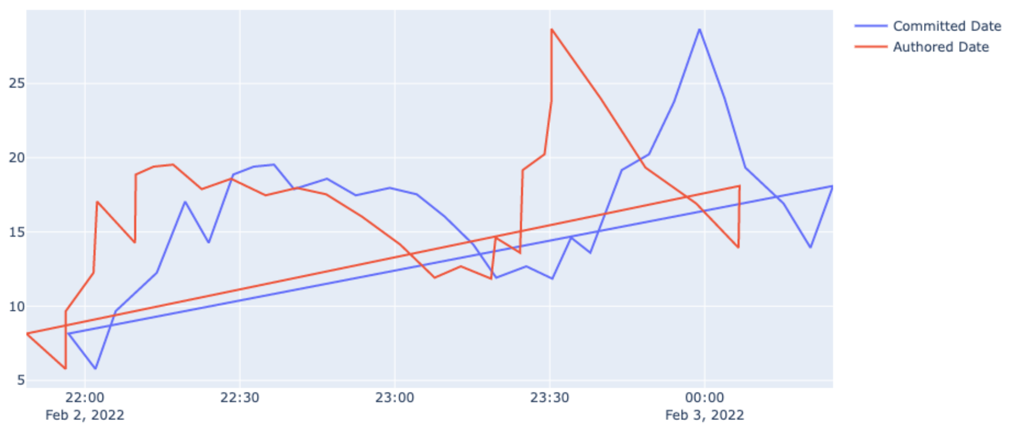
Top Pull Requests

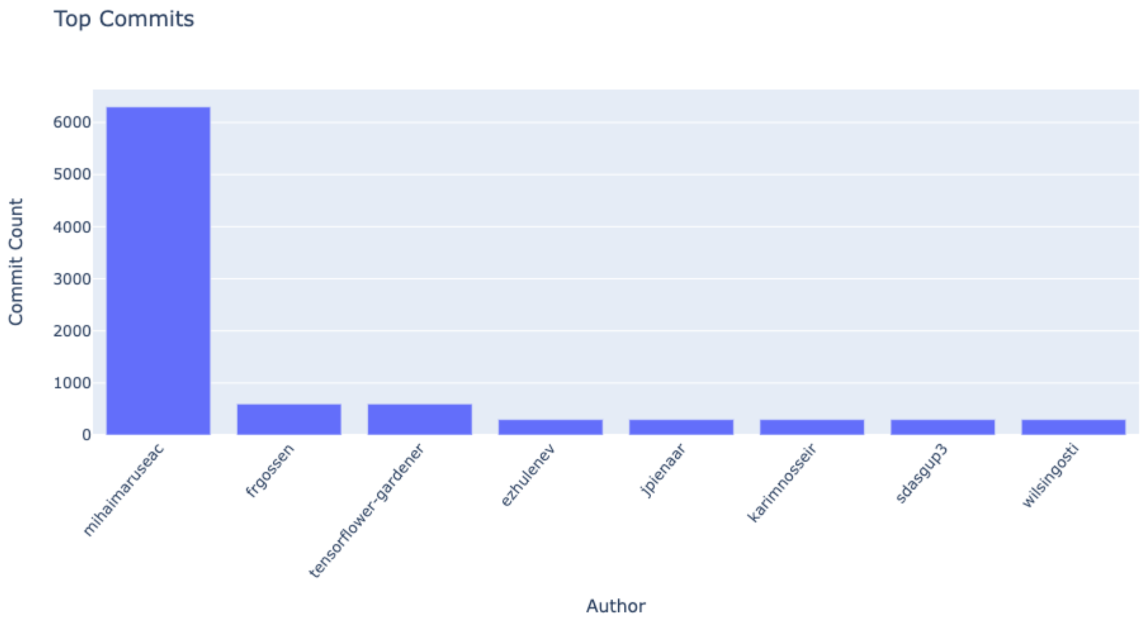


Commits

During the period of Feburary 2nd 2022 to Feburary 3rd 2022, there are 30 unique commits.

The average response time in the Tensorflow community is around 16 minutes, for the committer to confirm the commits. The top committer was *mihaimaruseac* with over 6000 counts.

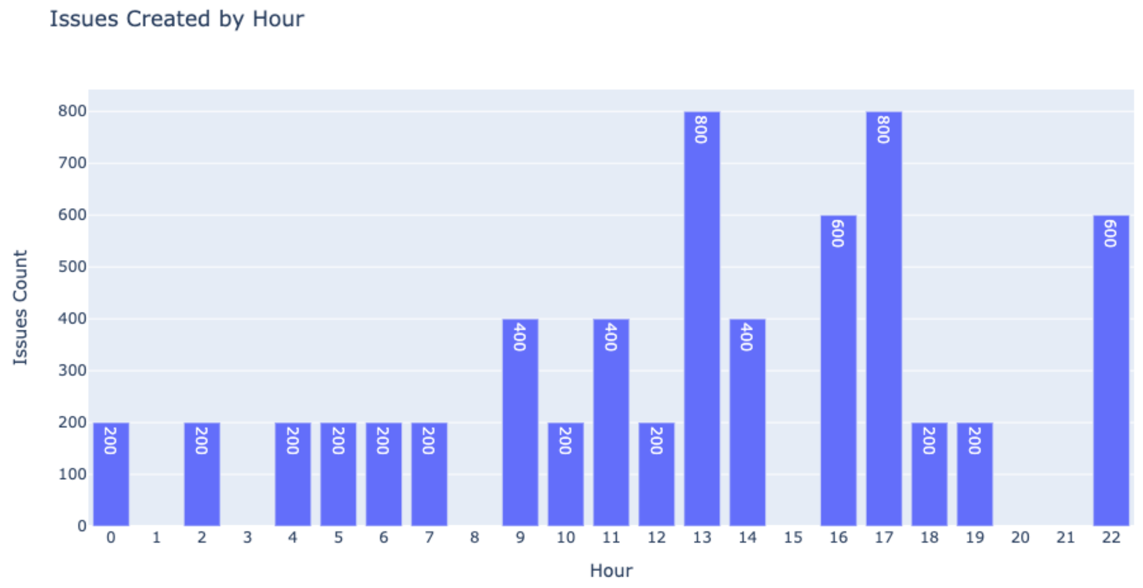


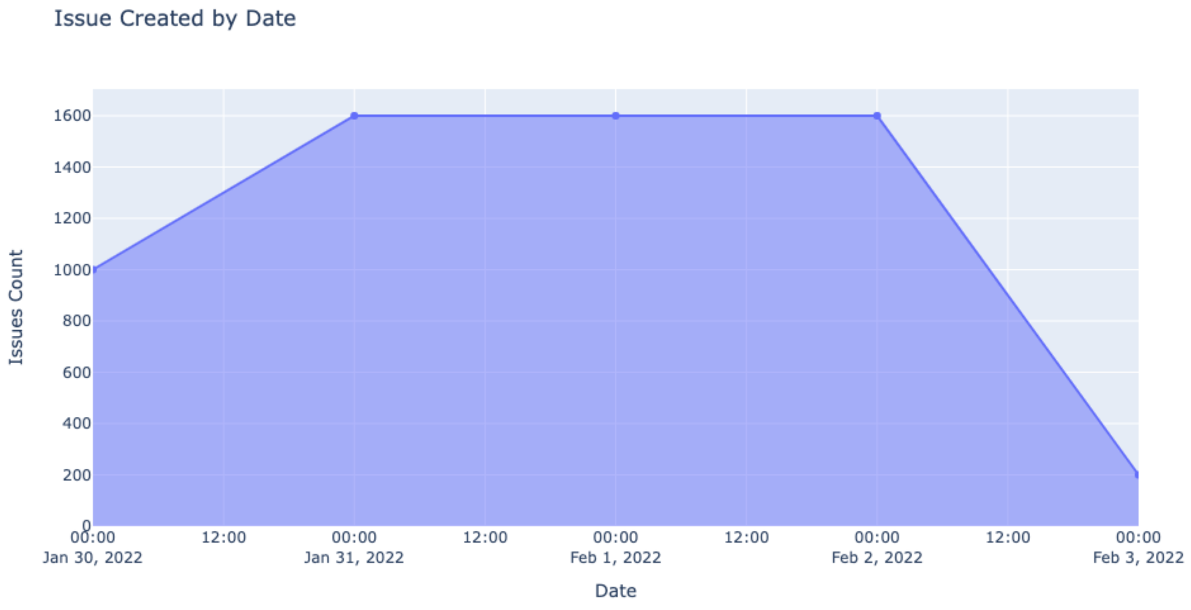


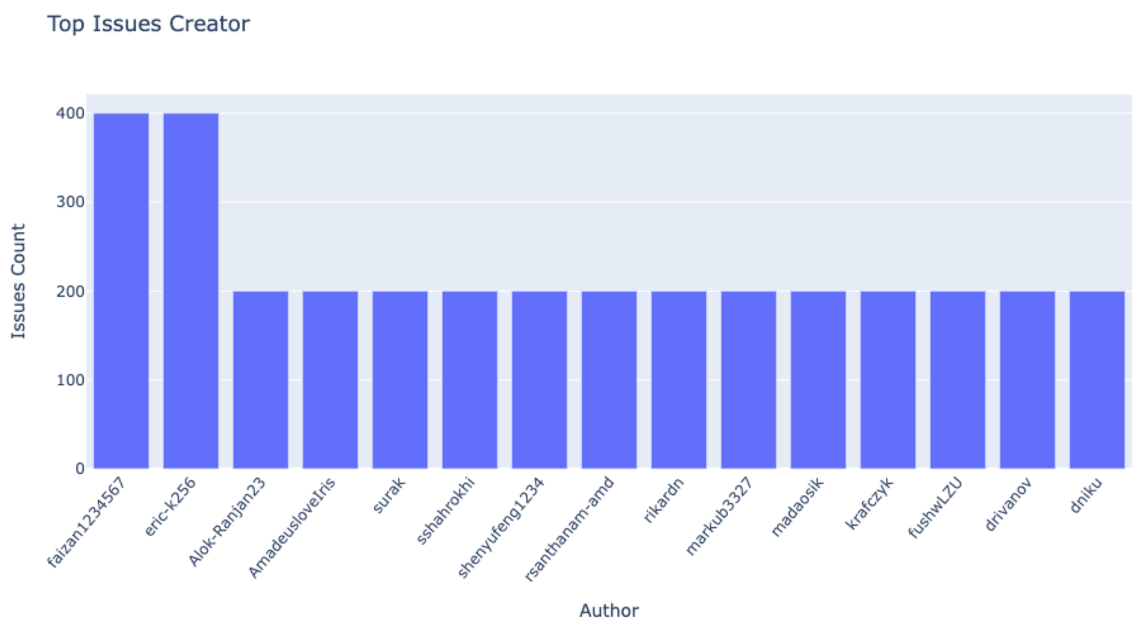
Issues

During the period of January 31th 2022 to Feburary 3rd 2022, there were 30 unique issues.

The top issue creator was *faizan1234567* and *eric-k256*with with each over 400 creations.
13 pm, 5 pm and 10 pm has the largest amount of issues being created.Starting Jan 30th, the issues creation rate became higher until Feb 3rd, which slowed down by 60%.

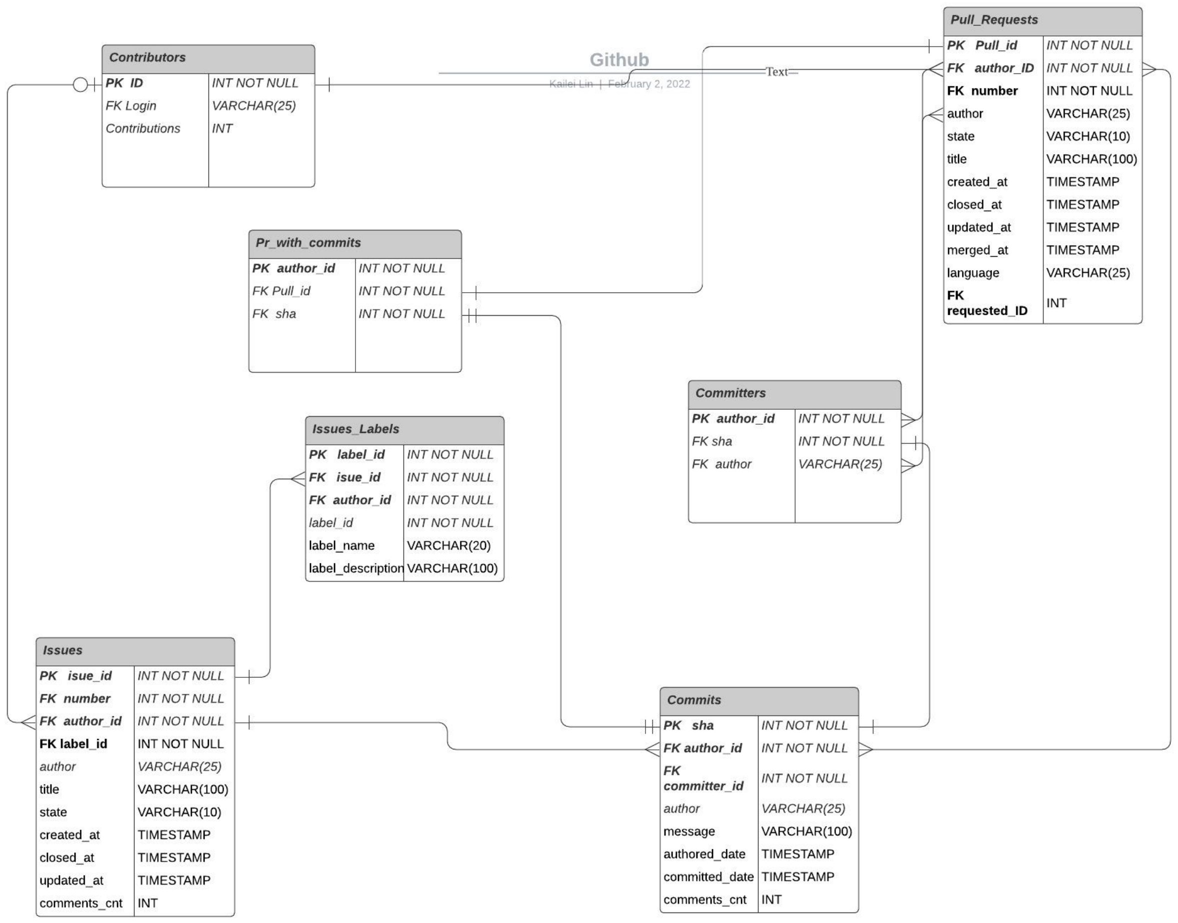






1. Table Schema Design

Given the API data and the project goal, in which monitoring KPIs to measure the active level of a repository, the database design should follow the RMDS normalization. *Pull Requests, Commits*, *Contributors*, and *Issues* are the main tables identified from Github official APIs. Each of the tables can be referenced by the mutual user account id, addition table was then normalized in order to further investigate the correlation between the size of an issue and the resolved time, for performance tracking.



Pull_Requests

PK Pull_id	INT NOT NULL
FK author_ID	INT NOT NULL
FK number	INT NOT NULL
author	VARCHAR(25)
state	VARCHAR(10)
title	VARCHAR(100)
created_at	TIMESTAMP
closed_at	TIMESTAMP
updated_at	TIMESTAMP
merged_at	TIMESTAMP
language	VARCHAR(25)
FK requested_ID	INT

Contributors

PK ID	INT NOT NULL
FK Login	VARCHAR(25)
Contributions	INT

Pr_with_commits

PK author_id	INT NOT NULL
FK Pull_id	INT NOT NULL
FK sha	INT NOT NULL

Committers

PK author_id	INT NOT NULL
FK sha	INT NOT NULL
FK author	VARCHAR(25)

Issues_Labels

PK label_id	INT NOT NULL
FK isue_id	INT NOT NULL
FK author_id	INT NOT NULL
label_id	INT NOT NULL
label_name	VARCHAR(20)
label_description	VARCHAR(100)

Issues

PK isue_id	INT NOT NULL
FK number	INT NOT NULL
FK author_id	INT NOT NULL
FK label_id	INT NOT NULL
author	VARCHAR(25)
title	VARCHAR(100)
state	VARCHAR(10)
created_at	TIMESTAMP
closed_at	TIMESTAMP
updated_at	TIMESTAMP
comments_cnt	INT

Commits

PK sha	INT NOT NULL
FK author_id	INT NOT NULL
FK committer_id	INT NOT NULL
author	VARCHAR(25)
message	VARCHAR(100)
authored_date	TIMESTAMP
committed_date	TIMESTAMP
comments_cnt	INT

1. ***Pull_Requests***

The pull requests table stores information about the pull requests within the repository, such as Pull request-id, author, request title, created date, closed date, merged date, current state, and language used.
Using the author id as Foreign key, the Pull Requests table can be joined by the following tables.

contributors
committers
commits
Issues
Pr_with_commits

2. ***Contributors***

The contributors table stores information about the contributor within the repository, such as contributor id, login name, and total number of contributions.
Using the author id as Foreign key, the contributors table can be joined by the following tables

Pull_Requests

Committers
Commits

3. Issues

The issues table stores information about the issues within the repository, such as issue id, author, issue title, created date, closed date, merged date, current state, and the number of comments.
Using the author id as Foreign key, the Pull Requests table can be joined by the following tables.

Issues_Labels
Pull_Requests
Commits
Contributors

4. Commits

The commits table stores information about the commits within the repository, such as sha, author, committer id, created date, closed date, committed date, and message.
Using the author id, sha as Foreign key, the Commits table can be joined by the following tables.

committers
Pull_Requests
Pr_with_commits

5. Issues_Labels

The issue_labels table stores information about the issue within the repository, such as issue id, issue name, issue description and author_id.
Using the author id, sha as Foreign key, the issue_labels table can be joined by the following tables.
Issues

6. Committers

The committers table stores information about the committers within the repository, such as author, committer id and sha.
Using the author id, sha as Foreign key, the Commits table can be joined by the following tables.
Contributors
Commits
Pull_requests

7. Pr_with_commits

The pr_with_commits table stores information about the pull requests that has been committed within the repository, such as sha, author, pull id.

Using the author id, sha as Foreign key, the Commits table can be joined by the following tables.
Pull_Requests
Commits
Committers

I. Normalized DBMS

Pros

1. Calculations are dynamic, based on the context of the scenarios
2. Faster dataset refresh time

Cons

1. Slower report performance

Indexing techniques can be used with the tables properly as mostly are read operations.

III. Integration with Development Tools

The schema design considers the integration with other issue tracking systems and tools, the *Commits, Files, and Developers* were selected. SHA identifiers, file names can be matched for *Commits* and *Files*.

IV. Query

1.average number of pull request per contributor?

```
SELECT b.Login, avg(CAST(b.pull_cnt AS FLOAT)) FROM(SELECT c.Login, c.ID, count(DISTINCTpull_id) as
pull_cnt from Pull_Requests p
INNER JOIN Contributors c
ON c.ID = p.author_ID
group by 1
order by 1) b as counts;
```

2. Maximum number of commits per week?

```
WITH cte AS (SELECT COUNT(DISTINCT SHA) as commits_cnt, WEEK(committed_date) as week FROM commits
group by week )
SELECT cte.week as Week,  MAX(cte.commits_cnt) as max_commits from cte
group by 1
order by 1 ;
```

3. Minimum number of days of an issue being open?

```
SELECT isue_id , DATEDIFF(closed_at, created_at) as open_day FROM issues group by 1
order by open_day asc;
```