BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Group Coursework Submission Form

Specialist Masters Programme

Please list all names of group members: (Surname, first name) 1.Ming Hao 2.Fanfei Chen 3.Yihong Ding	4.Hanrui Jiang 5. 6. 7. **GROUP NUMBER:** **8**
MSc in: Actuarial Science (1, 2), Actuarial Management (3, 4)	
Module Code: SMM695	
Module Title: Data Management System	
Lecturer: Matteo Devigili	**Submission Date: 22/07/2022**

Declaration:

By submitting this work, we declare that this work is entirely our own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the coursework instructions and any other relevant programme and module documentation. In submitting this work we acknowledge that we have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. We also acknowledge that this work will be subject to a variety of checks for academic misconduct.

We acknowledge that work submitted late without a granted extension will be subject to penalties, as outlined in the Programme Handbook. Penalties will be applied for a maximum of five days lateness, after which a mark of zero will be awarded.

Marker's Comments (if not being marked on-line):

Deduction for Late Submission:

Final Mark: %

Part 0: Choice of DBMS & environments

For our report, we chose PostgreSQL as our DBMS. For the convenience of importing ‘gitIssues.csv’ & ‘gitData.csv’, we used Psycopg2 in Python.

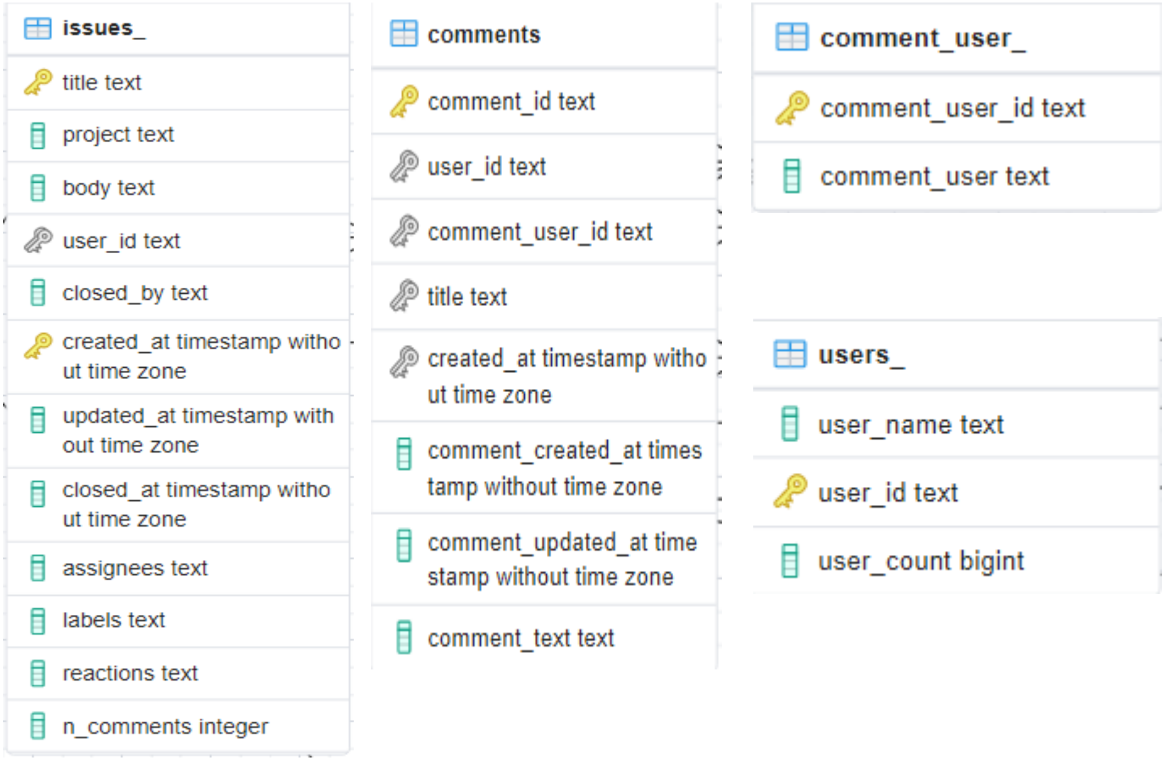
Part I: Justification of Design Choices

Firstly, we wrote an API python file to define classes and dictionaries for the main file we would use. (Note: Please run our API file first, named: ClassesAndDicts).

We defined two classes: ‘Schema’ and ‘Tables’ with functions to manipulate our database, eg: ‘drop_table_cascade’, etc. Two dictionaries are defined: ‘git_issues_data_dict’, ‘git_data_data_dict’ to help to import our data as tables in the database later in our main file.

There are two datasets provided: ‘gitIssues’ & ‘gitData’, which correspond to the history of Issues and Commits of Pytorch & TensorFlow. First, we imported these two datasets as DataFrames by using Pandas library. Then, we explored the dataset in python and made decisions as follows.

For issues data: ‘state’ is always closed, so we did not import this column into any tables in Issues Schema. We noticed that each issue has several comments made by the same user; therefore, we divided the ‘gitIssues’ dataset into four tables: ‘issues_’, ‘comments’, ‘comment_user_’, ‘users_’.



issues_
🔑 title text
project text
body text
🗝 user_id text
closed_by text
🔑 created_at timestamp without time zone
updated_at timestamp without time zone
closed_at timestamp without time zone
assignees text
labels text
reactions text
n_comments integer

comments
🔑 comment_id text
🗝 user_id text
🗝 comment_user_id text
🗝 title text
🗝 created_at timestamp without time zone
comment_created_at timestamp without time zone
comment_updated_at timestamp without time zone
comment_text text

comment_user_
🔑 comment_user_id text
comment_user text

users_
user_name text
🔑 user_id text
user_count bigint

For the table ‘issues_’, each row in this table corresponds to a specific issue reported. We chose all columns from the dataset ‘gitIssues’ and deleted all duplicated rows. The data types and primary key are set as follows:

- title, project, body, user_id, closed_by, assignees, labels, reactions: *text*
- created_at, updated_at, closed_at: *timestamp without time zone*
- n_comments: *integer*
- (title, created_at) is set as the composite primary key. We ensured the rows are unique for the composition of these two columns. Each specific issue can be identified by the title and created date since the title is typed by the user the date is accurate to seconds.

For the table ‘comments’, each row in this table corresponds to a specific comment made for issues. We chose all columns from the dataset ‘gitIssues’. The data types and primary key are set as follows:

- comment_id, user_id, comment_user_id, title, comment_text: *text*
- created_at, comment_created_at, comment_updated_at: *timestamp without time zone*
- comment_id is set as the primary key. We ensured the rows are unique for this column. Each specific comment can be identified by comment_id.

For the table ‘users_’, each row in this table corresponds to a specific user. We chose all columns from the dataset ‘gitIssues’ and deleted all duplicated rows. An additional column ‘user_count’ is added which represents the number of comments made by users. The data types and primary key are set as follows:

- user_name, user_id: *text*
- user_count: *bigint*, since this is made by using SQL.
- user_id is set as the primary key. We ensured the rows are unique for this column. Each specific user can be identified by user_id.

For the table ‘comment_user_’, each row in this table corresponds to a specific commenting user. We chose all columns from the dataset ‘gitIssues’. The data types and primary key are set as follows:

- comment_user_id, comment_user: *text*
- comment_user_id is set as the primary key. We ensured the rows are unique for this column. Each specific commenting user can be identified by comment_user_id.

Then, we built up an entity relationship and improve data integrity by setting foreign key constraints. The foreign key must refer to the primary key of another table and be consistent with the data of the referred primary key. Thus we added foreign key constraints as follows:

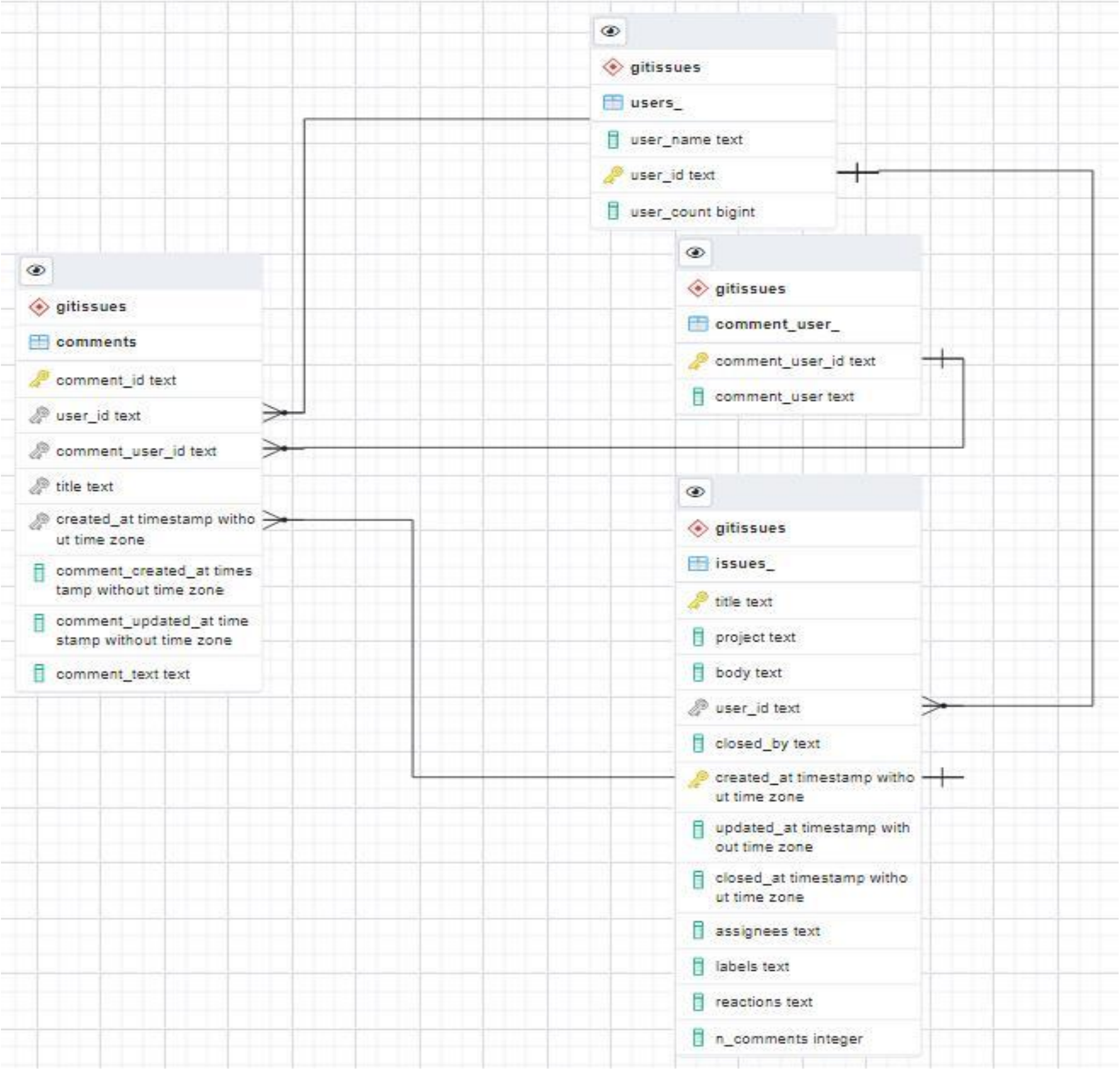
- We set user_id as the foreign key in ‘issues_’ refers to ‘users_’. This is because user_id is the primary key in ‘users_’, and user_id in both tables means the unique identifier of the user.

- We set user_id as the first foreign key in ‘comments’ refers to ‘users_’. This is because user_id is the primary key in ‘users_’, and user_id in both tables means the unique identifier of the user.

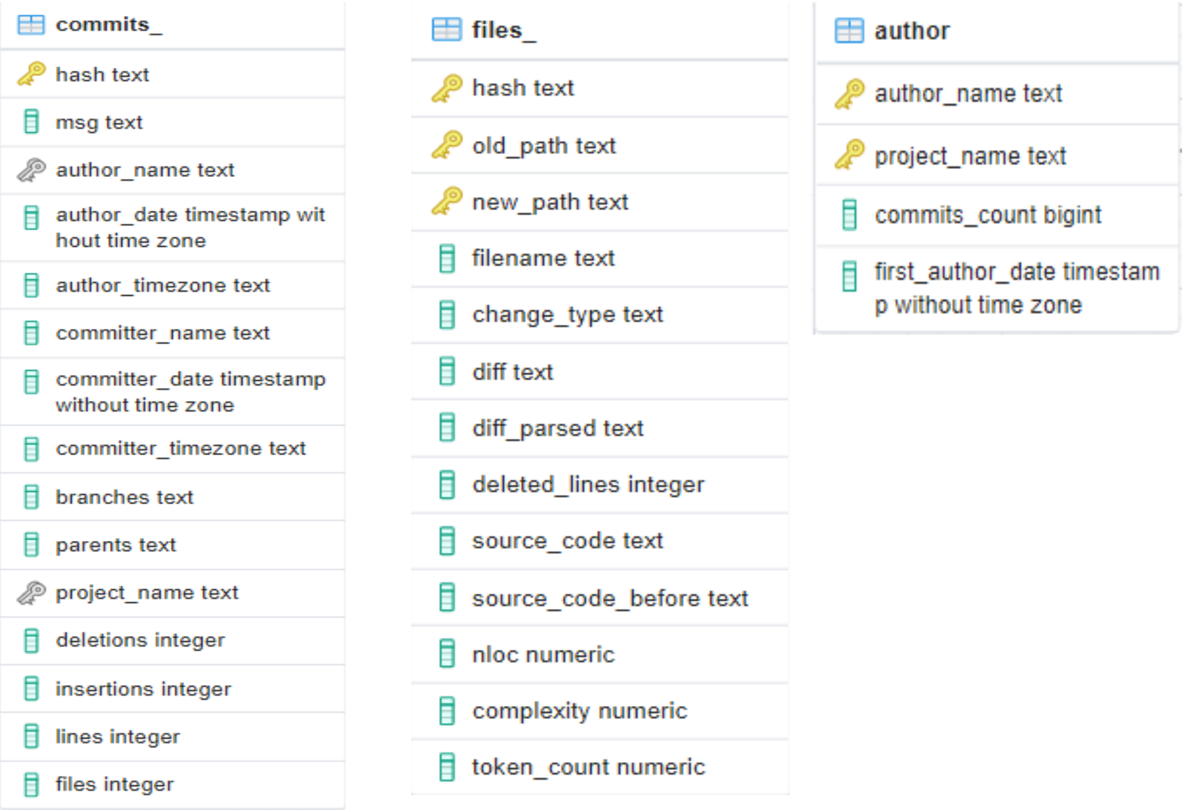
- We set (title, created_at) as the second (composite) foreign key in ‘comments’ refers to ‘issues_’. This is because (title, created_at) is the composite primary key in ‘issues_’, and (title, created_at) in both tables refers to a specific issue.

- We set comment_user_id as the third foreign key in ‘comments’ refers to ‘comment_user_’. This is because comment_user_id is the primary key in ‘comment_user’, and comment_user_id in both tables refers to a specific commenting user.

The final ER diagram for gitIssues schema is as below:



For commits data: ‘in_main_branch’ is always true, ‘_merge’ is always false. So we did not import these columns into any tables in gitData Schema. We noticed that each commit has several files made by the same author. Thus, we leveraged the ‘gitData’ dataset to make three tables: ‘commits_’, ‘files_’, ‘author’.



commits_
🔑 hash text
msg text
🔑 author_name text
author_date timestamp without time zone
author_timezone text
committer_name text
committer_date timestamp without time zone
committer_timezone text
branches text
parents text
🔑 project_name text
deletions integer
insertions integer
lines integer
files integer

files_
🔑 hash text
🔑 old_path text
🔑 new_path text
filename text
change_type text
diff text
diff_parsed text
deleted_lines integer
source_code text
source_code_before text
nloc numeric
complexity numeric
token_count numeric

author
🔑 author_name text
🔑 project_name text
commits_count bigint
first_author_date timestamp without time zone

For the table ‘commits_’, each row in this table corresponds to a specific commit. We chose all columns from the dataset ‘gitData’ and deleted all duplicated rows. The data types and primary key are set as follows:

- hash, msg, author_name, committer_name, author_timezone, committer_timezone, branches, parents, project_name: *text*
- author_date, committer_date: *timestamp without time zone*
- deletions, insertions, lines, files: *integer*
- hash is set as the primary key. We ensured the rows are unique for the composition of this column. Each specific issue can be identified by hash since hash can be regarded as commit id.

For the table ‘files’, each row in this table corresponds to a specific file committed. We chose all columns from the dataset ‘gitData’ and deleted all duplicated rows. The data types and primary key are set as follows:

- hash, old_path, new_path, filename, change_type, diff, diff_parsed, source_code, source_code_before: *text*
- nloc, complexity, token_count: *numeric* as they are decimals
- deleted_lines: integer
- (hash, old_path, new_path) is set as the composite primary key. For this table, it is hard to choose the pk since one hash can respond to several files, also

paths & filename can be the same for different commits. Finally, we choose (hash, old_path, new_path), And, we ensured the rows are unique for these columns, each specific file can be identified by (hash, old_path, new_path).

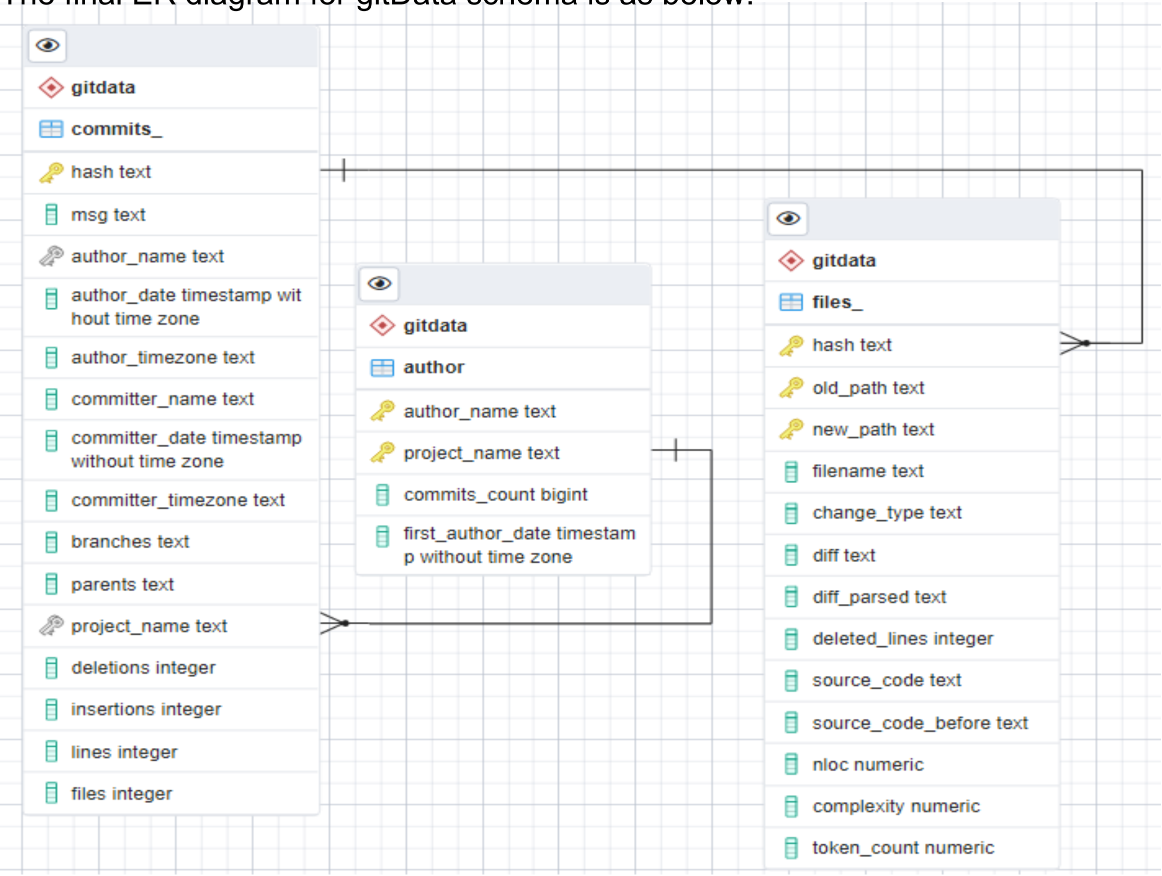
For the table ‘author’, each row in this table corresponds to a specific author for a project. We chose all columns from the dataset ‘gitData’ and deleted all duplicated rows. Additional columns ‘commits_count’ ‘first_author_date’ are added which represent the number of commits made by authors and the time of the first creation. The data types and primary key are set as follows:

- author_name, project_name: *text*
- commits_count: *bigint*, since this is made by using SQL.
- first_author_date: *date timestamp without time zone*
- (author_name, project_name) is set as the primary key. We ensured the rows are unique for these columns. Each specific author for a project can be identified by (author_name, project_name).

Then, we built up an entity relationship and improve data integrity by setting foreign key constraints:

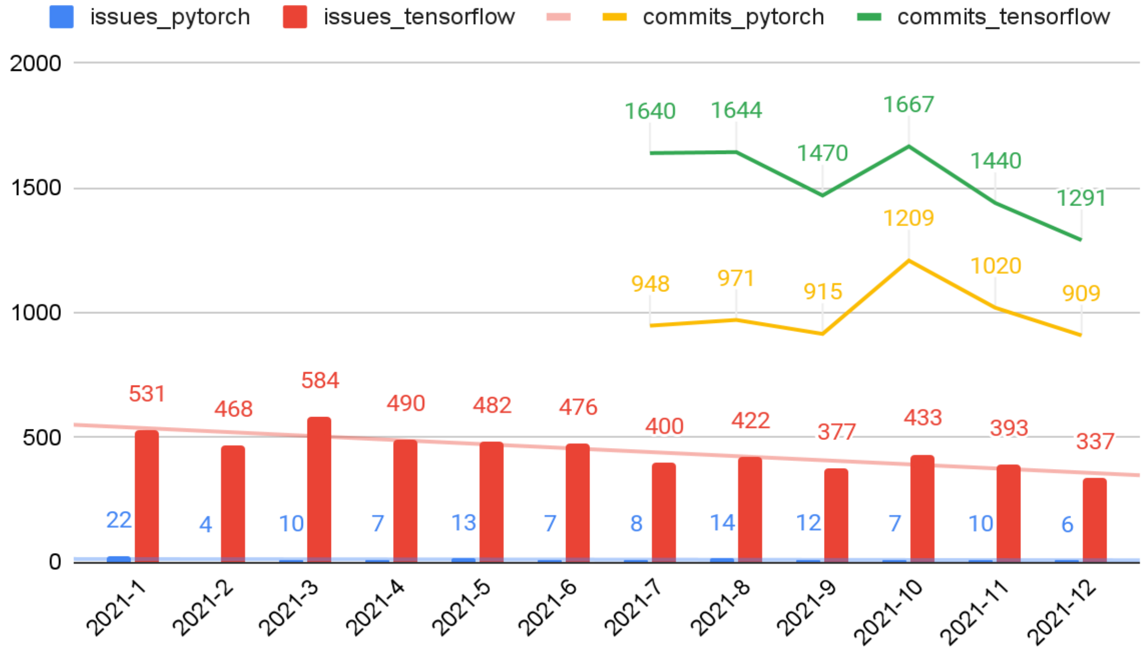
- We set hash as the foreign key in ‘files_’ refers to ‘commits_’. This is because hash is the primary key in ‘commits_’, and hash in both tables means the unique identifier of a commit.
- We set (author_name, project_name) as the foreign key in ‘commits’ refers to ‘author’. This is because (author_name, project_name) is the composite primary key in ‘author’, and (author_name, project_name) in both tables refers to a specific author for a project.

The final ER diagram for gitData schema is as below:



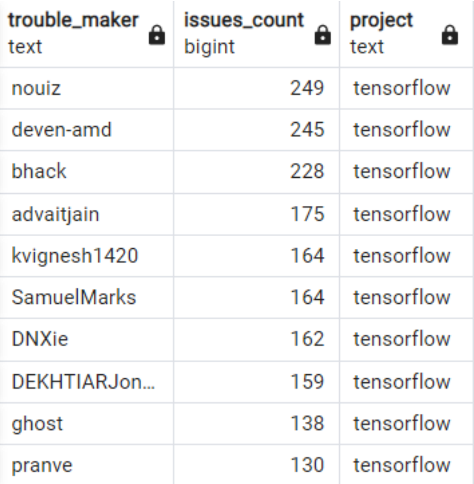
Part II: Description of the view insights

1. The first view insight is to explore new Issues and Commits numbers variation along the period. We select ***update time, project,*** and ***count*** from *gitIssues.issues_* table, and ***author time, project,*** and ***count*** from the *gitData.commits_* table, and use the 'Extract' function to get year-month from the timestamp.

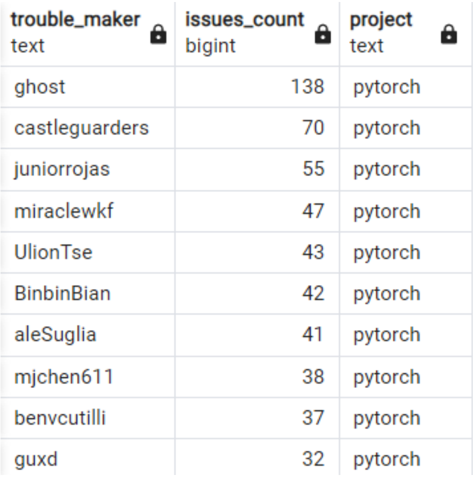


From the figure, we can see clearly that the issues reported declined steadily over the year 2021. This is because as the projects mature, problems were gradually solved. Meanwhile, for both 2 projects, the commits number peaked in 2021-10. We checked an article on Github (https://github.com/louisfb01/best_AI_papers_2021) and found that about 10 best AI papers are published around October. This might be one of the reasons why the number of commits for these two machine learning projects peaked in October.

2. The second view insight is to explore the Top10 Issues reporter('Trouble maker') and Commits creator('Contributor') for two projects over the period. We select ***user name, project,*** and ***count*** from (*gitIssues.issues_* inner join gitIssues.user), and ***author name, project,*** and ***count*** from the *gitData.commits_* table.

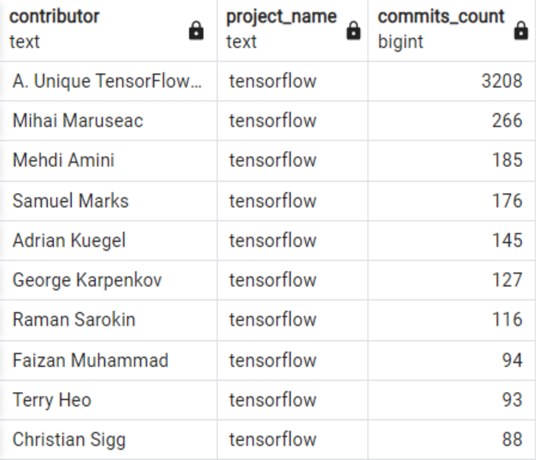


trouble_maker text	issues_count bigint	project text
nouiz	249	tensorflow
deven-amd	245	tensorflow
bhack	228	tensorflow
advaitjain	175	tensorflow
kvignesh1420	164	tensorflow
SamuelMarks	164	tensorflow
DNXie	162	tensorflow
DEKHTIARJon...	159	tensorflow
ghost	138	tensorflow
pranve	130	tensorflow

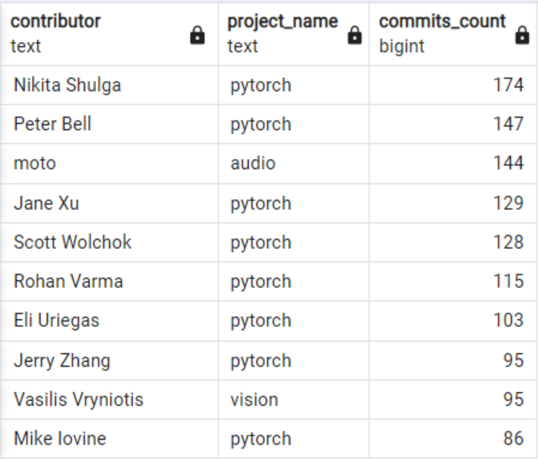


trouble_maker text	issues_count bigint	project text
ghost	138	pytorch
castleguarders	70	pytorch
juniorrojas	55	pytorch
miraclewkf	47	pytorch
UlionTse	43	pytorch
BinbinBian	42	pytorch
aleSuglia	41	pytorch
mjchen611	38	pytorch
benvcutilli	37	pytorch
guxd	32	pytorch

The user named ‘nouiz’ won the champion for ‘trouble maker’ for tensor flow with 249 issues, and the user ‘ghost’ won the one for PyTorch with 138 issues.



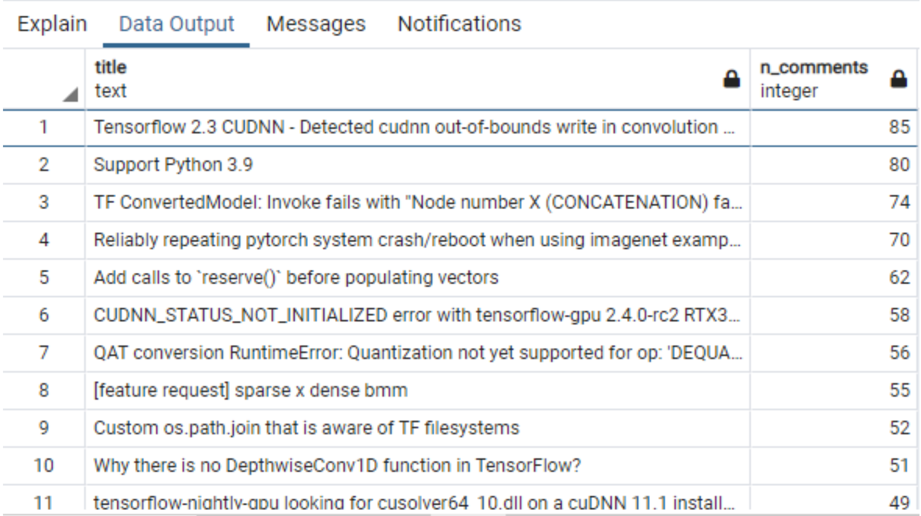
contributor text	project_name text	commits_count bigint
A. Unique TensorFlow...	tensorflow	3208
Mihai Maruseac	tensorflow	266
Mehdi Amini	tensorflow	185
Samuel Marks	tensorflow	176
Adrian Kuegel	tensorflow	145
George Karpenkov	tensorflow	127
Raman Sarokin	tensorflow	116
Faizan Muhammad	tensorflow	94
Terry Heo	tensorflow	93
Christian Sigg	tensorflow	88



contributor text	project_name text	commits_count bigint
Nikita Shulga	pytorch	174
Peter Bell	pytorch	147
moto	audio	144
Jane Xu	pytorch	129
Scott Wolchok	pytorch	128
Rohan Varma	pytorch	115
Eli Uriegas	pytorch	103
Jerry Zhang	pytorch	95
Vasilis Vryniotis	vision	95
Mike Iovine	pytorch	86

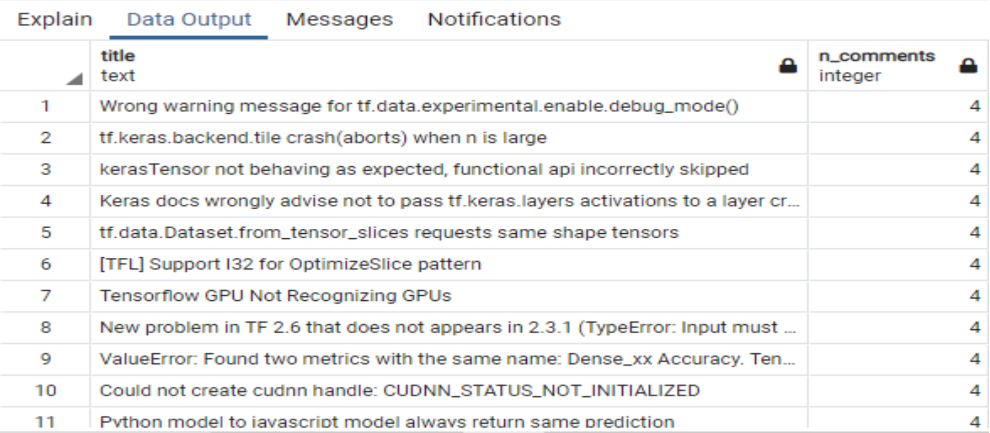
The author named ‘A. Unique TensorFlower’ won the champion for ‘contributor’ for tensor flow, this is perhaps an organization or bot with 3208 commits. And, the author ‘Nikita Shulga’ won the one for PyTorch with 174 issues.

3. The third view insight is created straightforwardly to find the most famous issue by ranking the number of comments for every title. By the descending ordering of n_comments, users can see the most famous issue has 85 comments.



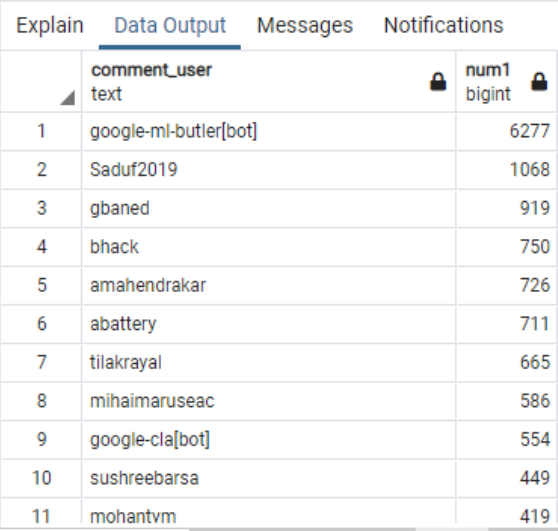
	title text	n_comments integer
1	Tensorflow 2.3 CUDNN - Detected cudnn out-of-bounds write in convolution ...	85
2	Support Python 3.9	80
3	TF ConvertedModel: Invoke fails with "Node number X (CONCATENATION) fa...	74
4	Reliably repeating pytorch system crash/reboot when using imagenet examp...	70
5	Add calls to `reserve()` before populating vectors	62
6	CUDNN_STATUS_NOT_INITIALIZED error with tensorflow-gpu 2.4.0-rc2 RTX3...	58
7	QAT conversion RuntimeError: Quantization not yet supported for op: 'DEQUA...	56
8	[feature request] sparse x dense bmm	55
9	Custom os.path.join that is aware of TF filesystems	52
10	Why there is no DepthwiseConv1D function in TensorFlow?	51
11	tensorflow-nightly-gpu looking for cusolver64_10.dll on a cuDNN 11.1 install...	49

In addition, more conditions can be added. For example, it can list the issues with comments less than 5 (by condition n_comments <5), and the total number of such issues is 2835. The first 11 rows of this view are shown below:

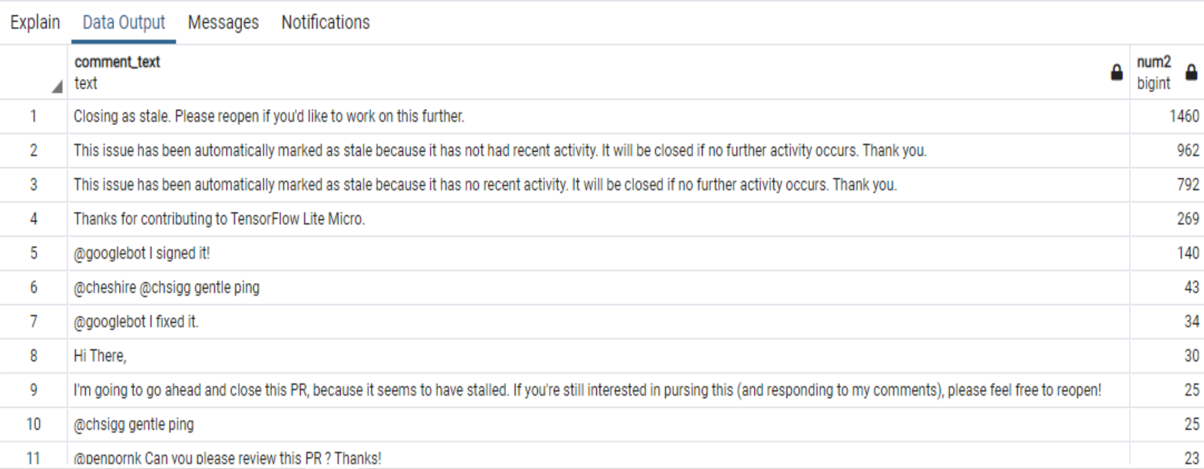


	title text	n_comments integer
1	Wrong warning message for tf.data.experimental.enable_debug_mode()	4
2	tf.keras.backend.tile crash(aborts) when n is large	4
3	kerasTensor not behaving as expected, functional api incorrectly skipped	4
4	Keras docs wrongly advise not to pass tf.keras.layers activations to a layer cr...	4
5	tf.data.Dataset.from_tensor_slices requests same shape tensors	4
6	[TFL] Support I32 for OptimizeSlice pattern	4
7	Tensorflow GPU Not Recognizing GPUs	4
8	New problem in TF 2.6 that does not appears in 2.3.1 (TypeError: Input must ...	4
9	ValueError: Found two metrics with the same name: Dense_xx Accuracy. Ten...	4
10	Could not create cudnn handle: CUDNN_STATUS_NOT_INITIALIZED	4
11	Python model to javascript model always return same prediction	4

4. The fourth view insight is generated to investigate the active users from the git.gitIssues. We select comment_user and count it as the number to form the basic table. By the descending order of the number of users, it is obvious that the google-ml-butler bot has the most comments with 6277; however, the second most active user “Saduf2019” is the real user. Similarly, we select comment_text to find out the most repeated comments in the same ways, indicating that the first four repeated comments are all produced by the google-ml-butler bot which is consistent with our result that the bot commented the most. The first 11 rows of this view are shown below:

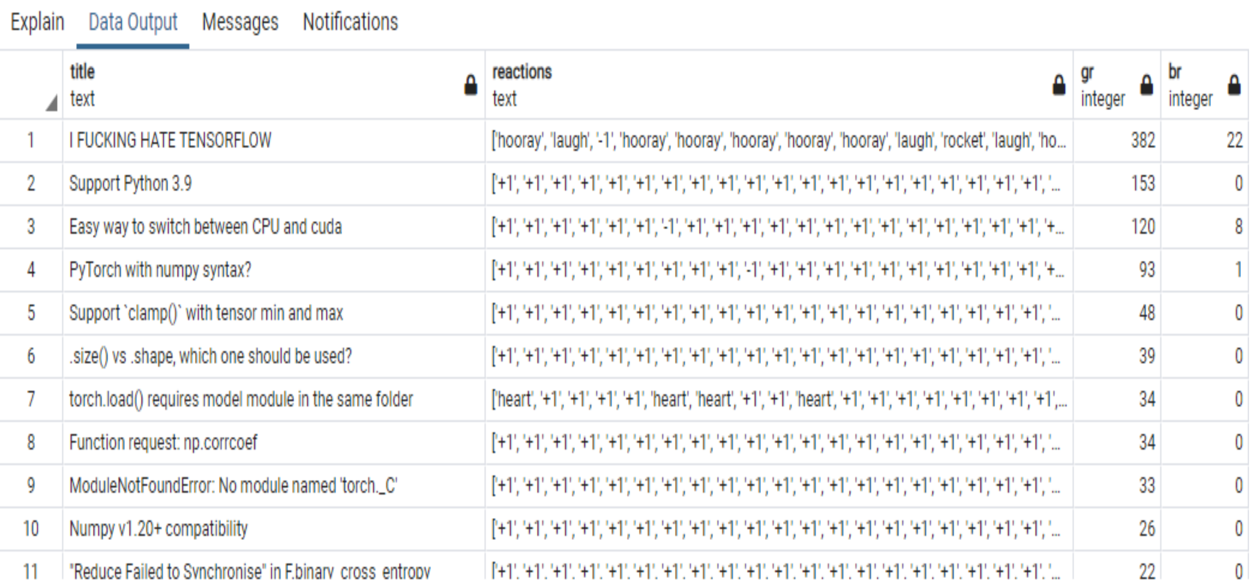


	comment_user text	num1 bigint
1	google-ml-butler[bot]	6277
2	Saduf2019	1068
3	gbaned	919
4	bhack	750
5	amahendrakar	726
6	abattery	711
7	tilakrayal	665
8	mihaimaruseac	586
9	google-cla[bot]	554
10	sushreebarsa	449
11	mohantym	419



	comment_text text	num2 bigint
1	Closing as stale. Please reopen if you'd like to work on this further.	1460
2	This issue has been automatically marked as stale because it has not had recent activity. It will be closed if no further activity occurs. Thank you.	962
3	This issue has been automatically marked as stale because it has no recent activity. It will be closed if no further activity occurs. Thank you.	792
4	Thanks for contributing to TensorFlow Lite Micro.	269
5	@googlebot I signed it!	140
6	@cheshire @chsigg gentle ping	43
7	@googlebot I fixed it.	34
8	Hi There,	30
9	I'm going to go ahead and close this PR, because it seems to have stalled. If you're still interested in pursing this (and responding to my comments), please feel free to reopen!	25
10	@chsigg gentle ping	25
11	@penpornk Can you please review this PR ? Thanks!	23

5. The fifth view insight is to explore the users’ reactions to the comments. Firstly, we select the title and reactions to form the basic table, and then divide the reactions into two types, one is the positive reactions(gr: including “+”, “laugh”, “hooray”, “heart”, “rocketr” and “eyes”), and the other is negative reactions(br: including “-” and “confused”), which need to be counted as the number to sort. By descending order of gr, the result is that the comment “I ******* HATE TENSORFLOW” has the most positive feedback but also has the highest number of bad reactions as well.



	title text	reactions text	gr integer	br integer
1	I FUCKING HATE TENSORFLOW	['hooray', 'laugh', '-1', 'hooray', 'hooray', 'hooray', 'hooray', 'hooray', 'laugh', 'rocket', 'laugh', 'ho...	382	22
2	Support Python 3.9	['+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '...	153	0
3	Easy way to switch between CPU and cuda	['+1', '+1', '+1', '+1', '+1', '+1', '-1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+...	120	8
4	PyTorch with numpy syntax?	['+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '-1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+...	93	1
5	Support `clamp()` with tensor min and max	['+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '...	48	0
6	.size() vs .shape, which one should be used?	['+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '...	39	0
7	torch.load() requires model module in the same folder	['heart', '+1', '+1', '+1', '+1', 'heart', 'heart', '+1', '+1', 'heart', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1',...	34	0
8	Function request: np.corrcoef	['+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '...	34	0
9	ModuleNotFoundError: No module named 'torch._C'	['+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '...	33	0
10	Numpy v1.20+ compatibility	['+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '...	26	0
11	"Reduce Failed to Synchronise" in F.binary_cross_entropy	['+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '+1', '...	22	0