Github.com

Database Management System
CSN - 351

Aman Saurabh	15114008
Amit Kumar	15114009
Nikhil Daf	15114021
Ravi Kumar	15114055
Sajal Sourav	15114062

INTRODUCTION :-

In this project we had to design and create the database of the popular website github.com, for doing that we were required :-

1. to develop the schema diagram and entity-relationship diagram then putting it in the form of tables.
2. normalization of the tables involved in the database.
3. filling data into the tables.
4. writing and running queries to verify the working of the database.

Basic entities in GitHub and functionalities between them:

- **User:**
  Users are personal GitHub accounts. Each user has a personal profile, and can own multiple repositories, public or private. They can create or be invited to join organizations or collaborate on another user's repository.

- **Project:**
  It contains Information about repositories. A repository is the most basic element of GitHub. They're easiest to imagine as a project's folder. A repository contains all of the project files (including documentation), and stores each file's revision history.

- **Project member:**
  Project member is a contributor to the project who is added by the admin of the project.
- **Commit:**
  A commit is an individual change to a file (or set of files). It's like when you save a file, except with Git, every time you save it creates a unique ID (a.k.a. the "SHA" or "hash") that allows you to keep record of what changes were made when and by who. Commits usually

contain a commit message which is a brief description of what changes were made.

- **Commit_comment:**

  It is the code review comments on commits made by github users.

- **Pull_request:**

  Pull requests are proposed changes to a repository submitted by a user and accepted or rejected by a repository's collaborators. Like issues, pull requests each have their own discussion forum.

- **Pull_request_history:**
  It represents events in the pull request lifetime.

- **Issue:**

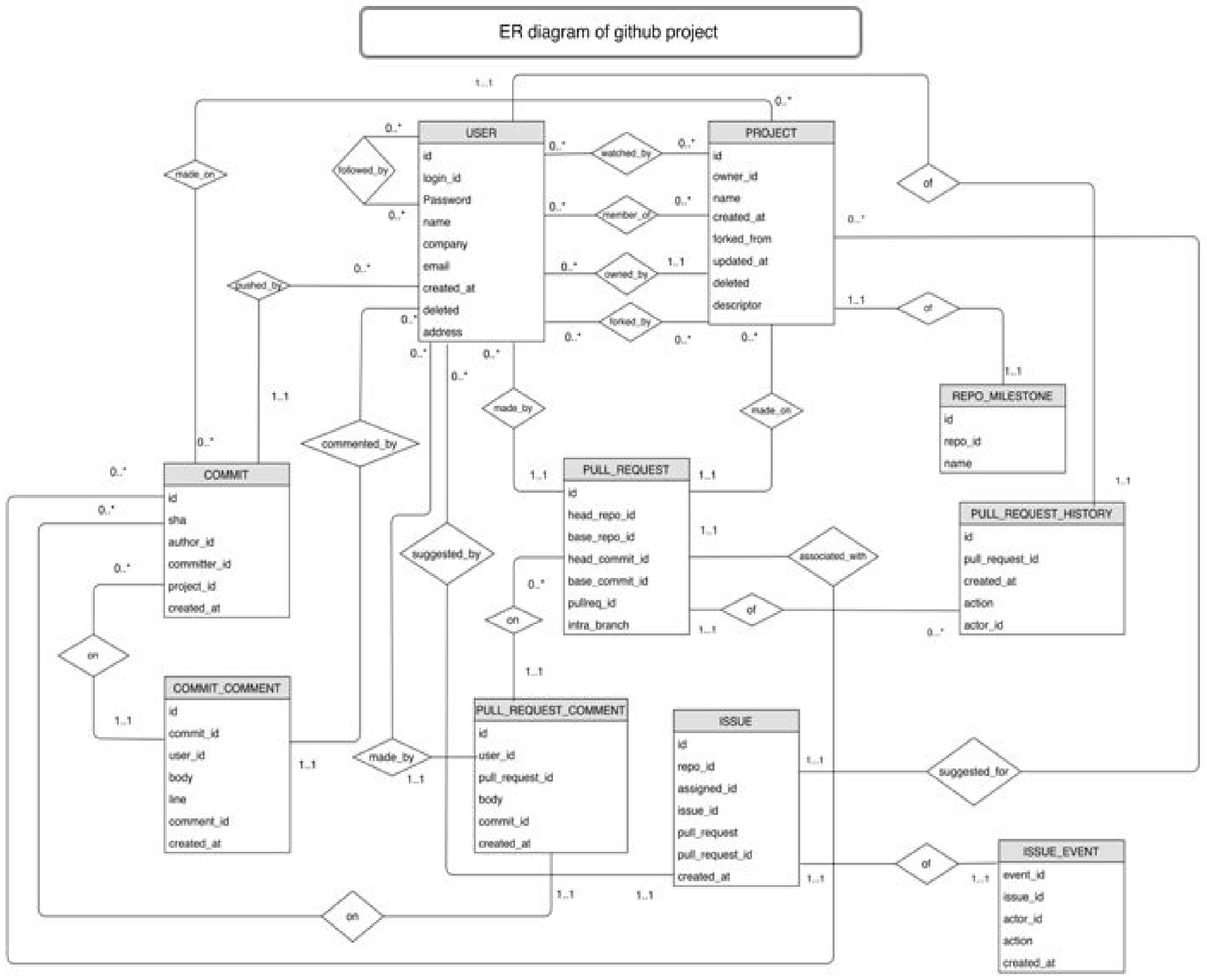
  Issues are suggested improvements, tasks or questions related to the repository. Issues can be created by anyone, and are moderated by repository collaborators. Each issue contains its own discussion forum, can be labeled and assigned to a user.

- **Issue_event:**
  It contains information about event actions on an issue related to a repository. Actions can be of many types.

- **Repo_milestone:**

  It keeps track of the repository updates.

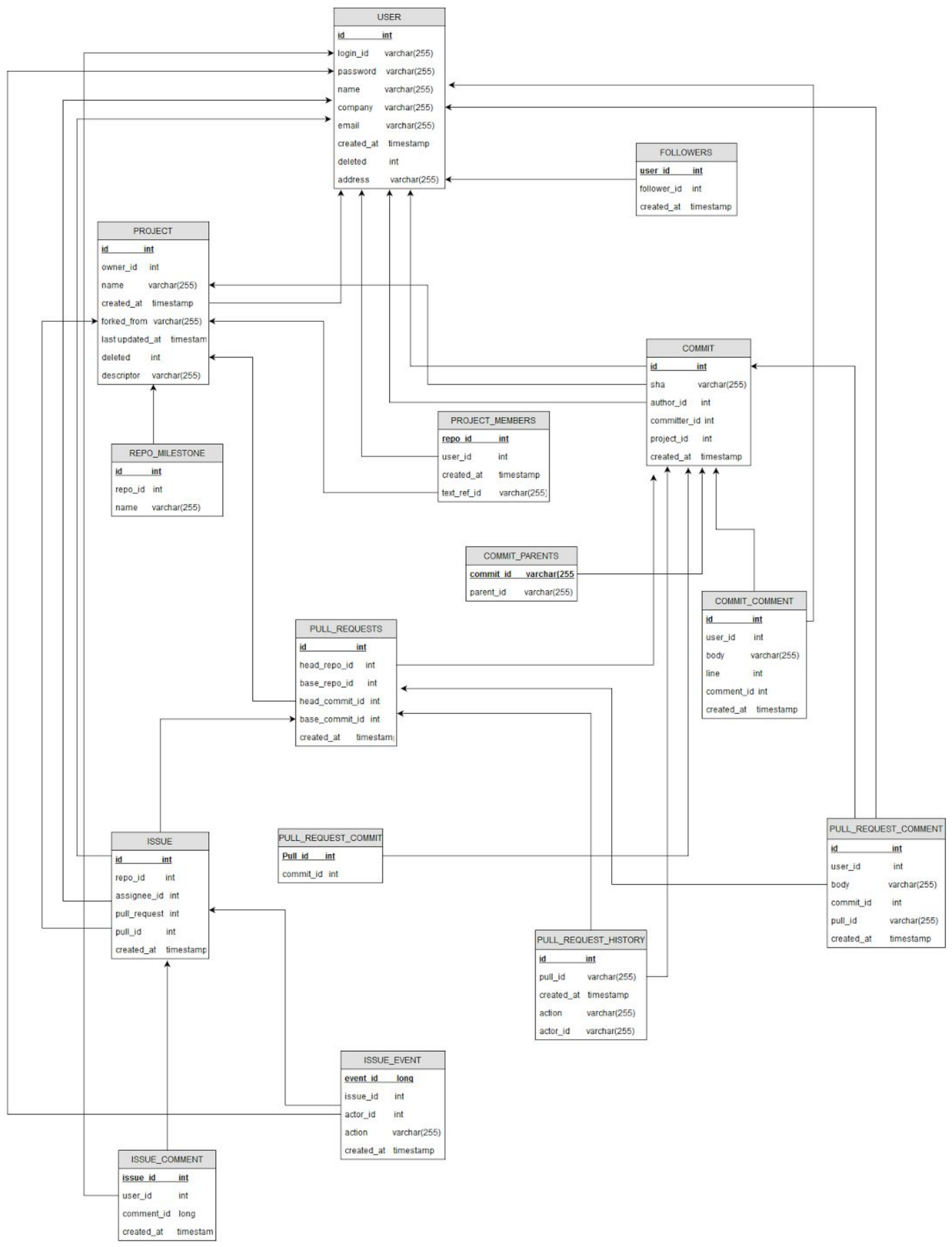
E-R DIAGRAM :-



ER diagram of github project
USER
id
login_id
Password
name
company
email
created_at
deleted
address
PROJECT
id
owner_id
name
created_at
forked_from
updated_at
deleted
descriptor
REPO_MILESTONE
id
repo_id
name
COMMIT
id
sha
author_id
committer_id
project_id
created_at
PULL_REQUEST
id
head_repo_id
base_repo_id
head_commit_id
base_commit_id
pullreq_id
intra_branch
PULL_REQUEST_HISTORY
id
pull_request_id
created_at
action
actor_id
COMMIT_COMMENT
id
commit_id
user_id
body
line
comment_id
created_at
PULL_REQUEST_COMMENT
id
user_id
pull_request_id
body
commit_id
created_at
ISSUE
id
repo_id
assigned_id
issue_id
pull_request
pull_request_id
created_at
ISSUE_EVENT
event_id
issue_id
actor_id
action
created_at
followed_by
watched_by
member_of
owned_by
forked_by
made_on
pushed_by
commented_by
made_by
made_on
of
of
suggested_by
associated_with
of
on
on
made_by
suggested_for
of
on
0..*
1..1

SCHEMA DIAGRAM :-

Schema diagram of github project



Functional Dependencies and normalization of corresponding tables:

User:

Attributes:

user_id, login_id, password, name, company, email, created_at, deleted, address.

Functional dependencies:

User_id -> login_id, password, name, company, email, created_at, deleted, address

Login_id, password -> user_id, name, company, email, created_at, deleted, address

Email -> user_id, login_id, password, name, company, created_at, deleted, address

Since the table is already in BCNF, no normalization is required.

Project:

Attributes:

project_id, owner_id, name, created_at, forked_from, last_updated_at, deleted, descriptor

Functional Dependencies:

Project_id -> owner_id, name, created_at, forked_from, last_updated_at, deleted, descriptor

Owner_id, name -> project_id, created_at, forked_from, last_updated_at, deleted, descriptor

Since the table is already in BCNF, no normalization is required.

Follower:

Attributes:

fid,user_id, follower_id, created_at

Functional Dependencies:

fid -> User_id,follower_id, created_at

Since the table is already in BCNF, no normalization is required.

Project_Member

Attributes:

repo_id, user_id, created_at

Functional Dependency:

Repo_id , user_id -> created_at

Since the table is already in BCNF, no normalization is required.

Commit

Attributes:

id, sha, committer_id, project_id, created_at

Functional Dependency:

id->sha, committer_id, project_id, created_at

Commit_comment:

Attribute:

commit_comment_id, commit_id, user_id, body, line, created_at

Functional Dependency:

commit_comment_id-> commit_id, user_id, body, line, created_at

Since the table is already in BCNF, no normalization is required.

Pull_requests:

Attributes:

pull_req_id, head_repo_id, base_repo_id,base_commit_id

Functional dependencies:

pull_req_id-> head_repo_id, base_repo_id,base_commit_id

base_repo_id-> head_repo_id

Normalized tables:

T1(pull_req_id, base_repo_id, base_commit_id)

T2(base_repo_id, head_repo_id)

Pull_request_commit:

Attributes:
pull_id, commit_id

Functional Dependency:

pull_id->commit_id

Since the table is already in BCNF, no normalization is required.

Pull_request_history:

Attributes:

pull_request_history_id, pull_request_id, created_at, action, actor_id

Functional Dependency:

pull_request_history_id-> pull_request_history_id, pull_request_id, created_at, action, actor_id

Since the table is already in BCNF, no normalization is required.

Pull_request_comment:

Attributes:

comment_id, user_id, body, pull_request_id, created_at

Functional Dependency:

comment_id->comment_id, user_id, body, pull_request_id, created_at

Issue:

Attribute:

issue_id, repo_id, issuer_id,assignee_id,pull_request_id, created_at

Functional Dependency:

issue_id->issue_id, repo_id, issuer_id, assignee_id, pull_request, pull_request_id, created_at

Since the table is already in BCNF, no normalization is required.

Issue event:

Attribute:event_id, issue_id, actor_id, action, created_at

Functional Dependency:

event_id->event_id, issue_id, action, created_at

issue_id->actor_id

Normalized tables:

T1(event_id, issue_id, action, created_at)

T2(issue_id,actor_id)

Issue_comment:

Attribute:

comment_id,issue_id, user_id, body , created_at

Functional Dependency:

comment_id->issue_id, user_id, body, created_at

Since the table is already in BCNF, no normalization is required.

CREATING AND FILLING TABLES :-

After the finally deciding the relevant tables, phpmyadmin was used for use to create the database followed by adding tables to it and filling data in the tables. Following shows the relevant information :-

Table entries :-

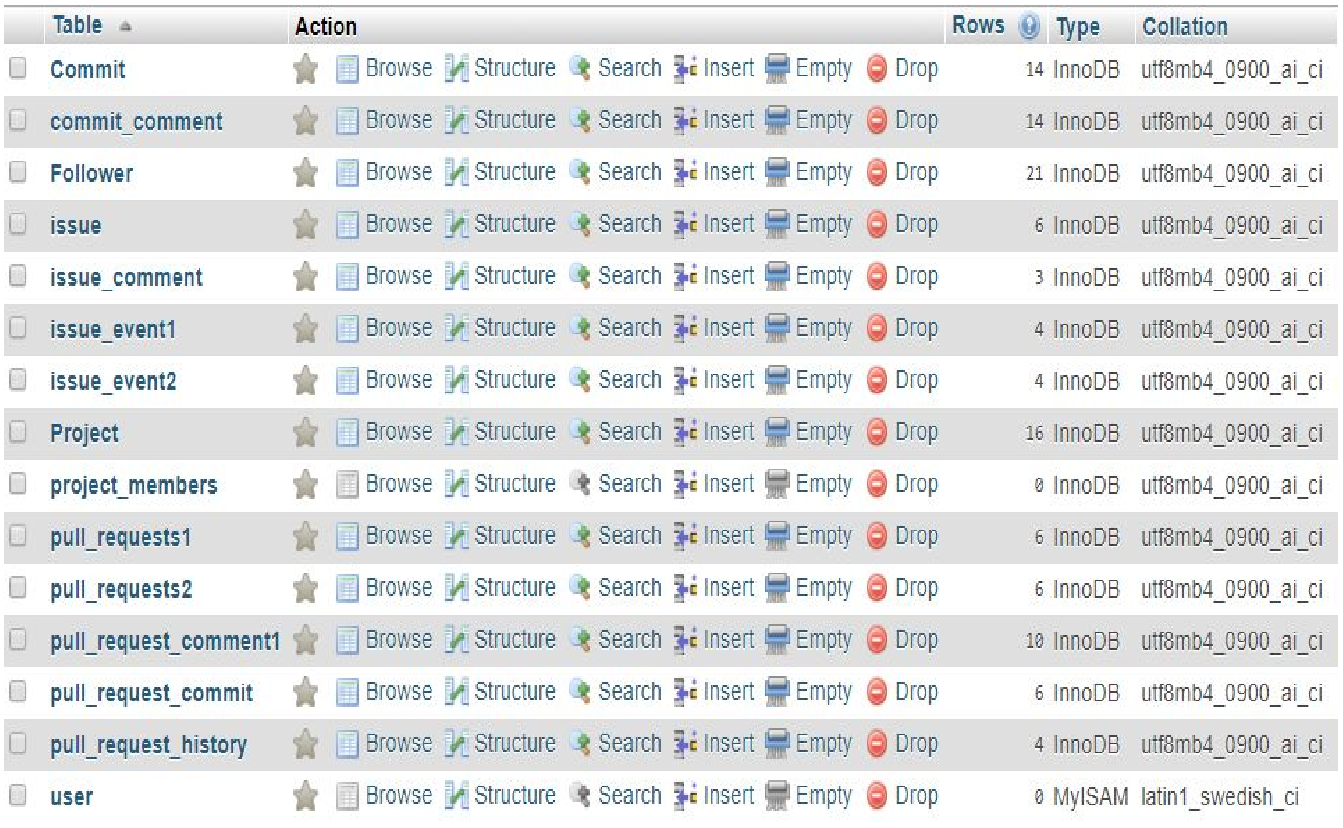
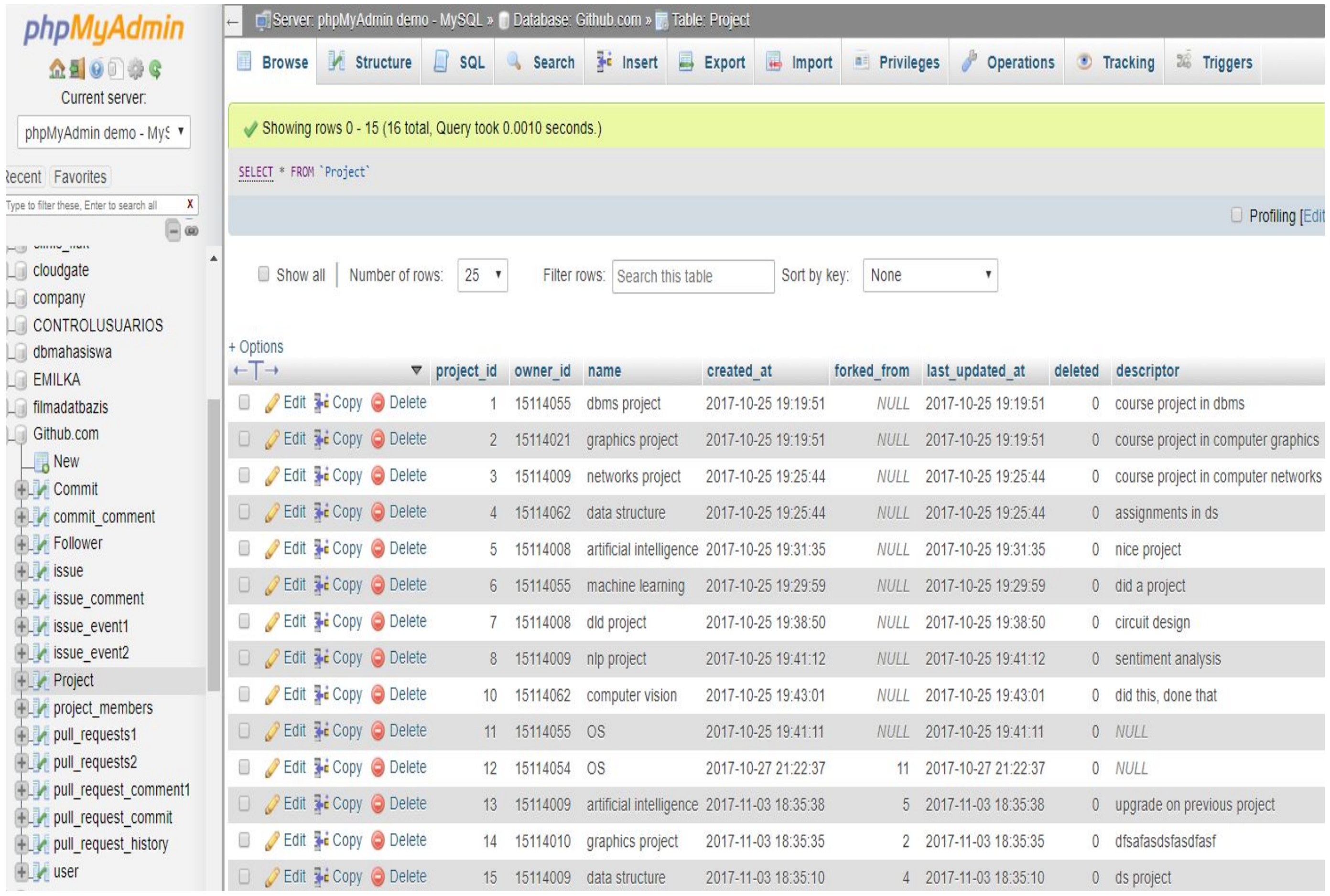


Table	Action	Rows	Type	Collation
Commit	Browse Structure Search Insert Empty Drop	14	InnoDB	utf8mb4_0900_ai_ci
commit_comment	Browse Structure Search Insert Empty Drop	14	InnoDB	utf8mb4_0900_ai_ci
Follower	Browse Structure Search Insert Empty Drop	21	InnoDB	utf8mb4_0900_ai_ci
issue	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_0900_ai_ci
issue_comment	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_0900_ai_ci
issue_event1	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_0900_ai_ci
issue_event2	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_0900_ai_ci
Project	Browse Structure Search Insert Empty Drop	16	InnoDB	utf8mb4_0900_ai_ci
project_members	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_0900_ai_ci
pull_requests1	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_0900_ai_ci
pull_requests2	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_0900_ai_ci
pull_request_comment1	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_0900_ai_ci
pull_request_commit	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_0900_ai_ci
pull_request_history	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_0900_ai_ci
user	Browse Structure Search Insert Empty Drop	0	MyISAM	latin1_swedish_ci

Filling Data in a table :-



Showing rows 0 - 15 (16 total, Query took 0.0010 seconds.)

```
SELECT * FROM `Project`
```

Number of rows: 25 · Filter rows: Search this table · Sort by key: None

project_id	owner_id	name	created_at	forked_from	last_updated_at	deleted	descriptor
1	15114055	dbms project	2017-10-25 19:19:51	NULL	2017-10-25 19:19:51	0	course project in dbms
2	15114021	graphics project	2017-10-25 19:19:51	NULL	2017-10-25 19:19:51	0	course project in computer graphics
3	15114009	networks project	2017-10-25 19:25:44	NULL	2017-10-25 19:25:44	0	course project in computer networks
4	15114062	data structure	2017-10-25 19:25:44	NULL	2017-10-25 19:25:44	0	assignments in ds
5	15114008	artificial intelligence	2017-10-25 19:31:35	NULL	2017-10-25 19:31:35	0	nice project
6	15114055	machine learning	2017-10-25 19:29:59	NULL	2017-10-25 19:29:59	0	did a project
7	15114008	dld project	2017-10-25 19:38:50	NULL	2017-10-25 19:38:50	0	circuit design
8	15114009	nlp project	2017-10-25 19:41:12	NULL	2017-10-25 19:41:12	0	sentiment analysis
10	15114062	computer vision	2017-10-25 19:43:01	NULL	2017-10-25 19:43:01	0	did this, done that
11	15114055	OS	2017-10-25 19:41:11	NULL	2017-10-25 19:41:11	0	NULL
12	15114054	OS	2017-10-27 21:22:37	11	2017-10-27 21:22:37	0	NULL
13	15114009	artificial intelligence	2017-11-03 18:35:38	5	2017-11-03 18:35:38	0	upgrade on previous project
14	15114010	graphics project	2017-11-03 18:35:35	2	2017-11-03 18:35:35	0	dfsafasdsfasdfasf
15	15114009	data structure	2017-11-03 18:35:10	4	2017-11-03 18:35:10	0	ds project

WRITING AND RUNNING QUERIES :-

To verify the implementation and working of the database, several queries related to that of functioning of github.com were written and tested on the database. Some of those queries are :-

1.Adding a user to the database :-

INSERT INTO `user` (`user_id`, `login_id`, `password`, `name`, `company`, `email`, `created_at`, `deleted`, `address`) VALUES (15114002, 'sjw', 'sajwan', 'Abhishek', 'Google ', 'abhisaj@gmail.com', '2017-11-003 22:00:00', 0, 'S-72 iit roorkee');

2. Delete a user from the database ?

UPDATE `user` SET `deleted` = `1` WHERE `user`.`user_id` = 15114002;

3.Fork any repository from project table in the database :-

INSERT INTO `Project` (`project_id`, `owner_id`, `name`, `created_at`, `forked_from`, `last_updated_at`, `deleted`, `descriptor`) VALUES

(18, 15114002, 'dld project', '2017-11-04 17:10:51', 7, '2017-11-04 17:19:51', 0, 'description');

4.Add any repository in the project table :-

INSERT INTO `Project` (`project_id`, `owner_id`, `name`, `created_at`, `forked_from`, `last_updated_at`, `deleted`, `descriptor`) VALUES

(20, 15114011, 'toc project', '2017-11-03 17:10:51', NULL, '2017-11-03 17:19:51', 0, 'description1');

5. Add a project member to any project ?

INSERT INTO `project_members`(`project_id`, `user_id`, `created_at`) VALUES(7, 15114055, '2017-11-04 00:10:51');

6.Create commit on any repository or project :-

INSERT INTO `Commit`(`id`, `sha`, `committer_id`, `created_at`, `project_id`) VALUES (16,’sjdshjkhdasfkjhj’,15114021,’2017-10-25 19:45:36’,12);

7.Insert an issue in a repository?

INSERT INTO `issue`(`id`, `repo_id`, `issuer_id`, `assignee_id`, `pull_id`, `created_at`) VALUES (11,15,15114055,15114009,7,’2017-10-25 14:45:36’)

8.Add a comment on an issue?

INSERT INTO `issue_comment`(`issue_comment_id`, `issue_id`, `commenter_id`, `body`, `created_at`) VALUES (4,5,15114008,’kdfjlaskdfj’,’2017-10-25 14:37:32’)

9.Update a comment on an issue :-

UPDATE `issue_comment` SET `issue_comment_id`=1,`issue_id`=1,`commenter_id`=15114008,`body`=’random text’,`created_at`=’2017-10-25 14:37:32’ WHERE issue_comment_id = 1

10.Delete an issue comment?

DELETE FROM `issue_comment` WHERE issue_comment_id = 3

11.Update a comment?

UPDATE `issue_comment` SET `issue_comment_id`=1,`issue_id`=1,`commenter_id`=15114008,`body`=’random text’,`created_at`=’2017-10-25 14:37:32’ WHERE issue_comment_id = 1

12.Add an issue event on an issue?

INSERT INTO `issue_event1`(`event_id`, `issue_id`, `action`, `created_at`) VALUES (5,3,’merged’,’2017-11-05 15:33:39’)

13.Add the actor of the issue to database?

INSERT INTO `issue_event2`(`issue_id`, `actor_id`) VALUES (5,15114009)

14.A user with user_id = 15114055 is followed by another user with user_id = 15114054 ?

INSERT INTO `Follower`(`fid`, `user_id`, `follower_id`, `created_at`) VALUES (22, 15114055,15114054,’2017-11-05 01:01:00’)

15.Update an issue:

UPDATE `issue` SET `repo_id`=14, `issuer_id`=15114021, `assignee_id`=15114010, `pull_id`=3, `created_at`= ‘2017-11-05 03:10:00’ where issue.id = 11

16.Insert a new entry in pull request comment?

INSERT INTO `pull_request_comment1`(`id`, `user_id`, `body`, `pull_id`, `created_at`) VALUES (11,15114021,’very good’,7,’2017-11-03 18:03:19’);

17.Search all user_id whose pull_id=4?

```
SELECT `user_id`

FROM `pull_request_comment1`

WHERE pull_id=4;
```

18. Find all projects whose member is user with user_id =15114055 ?

```
SELECT Project.*
from project_members,Project
where Project.project_id=project_members.project_id and
project_members.user_id = 15114055;
```

19.print body of pull request comment whose user_id =15114021?

```
        SELECT  `body`
         FROM `pull_request_comment1`
        WHERE user_id=15114021;
```

20.update the action taken on 2017-11-04 00:11:19 from closed to merge?

```
        UPDATE `pull_request_history` SET
        `id`=4,`pull_id`=3,`created_at`='2017-11-04
        00:11:19',`action`=merged,`action_id`=2 WHERE created
        at='2017-11-04 00:11:19';
```