NYU Tandon School of Engineering

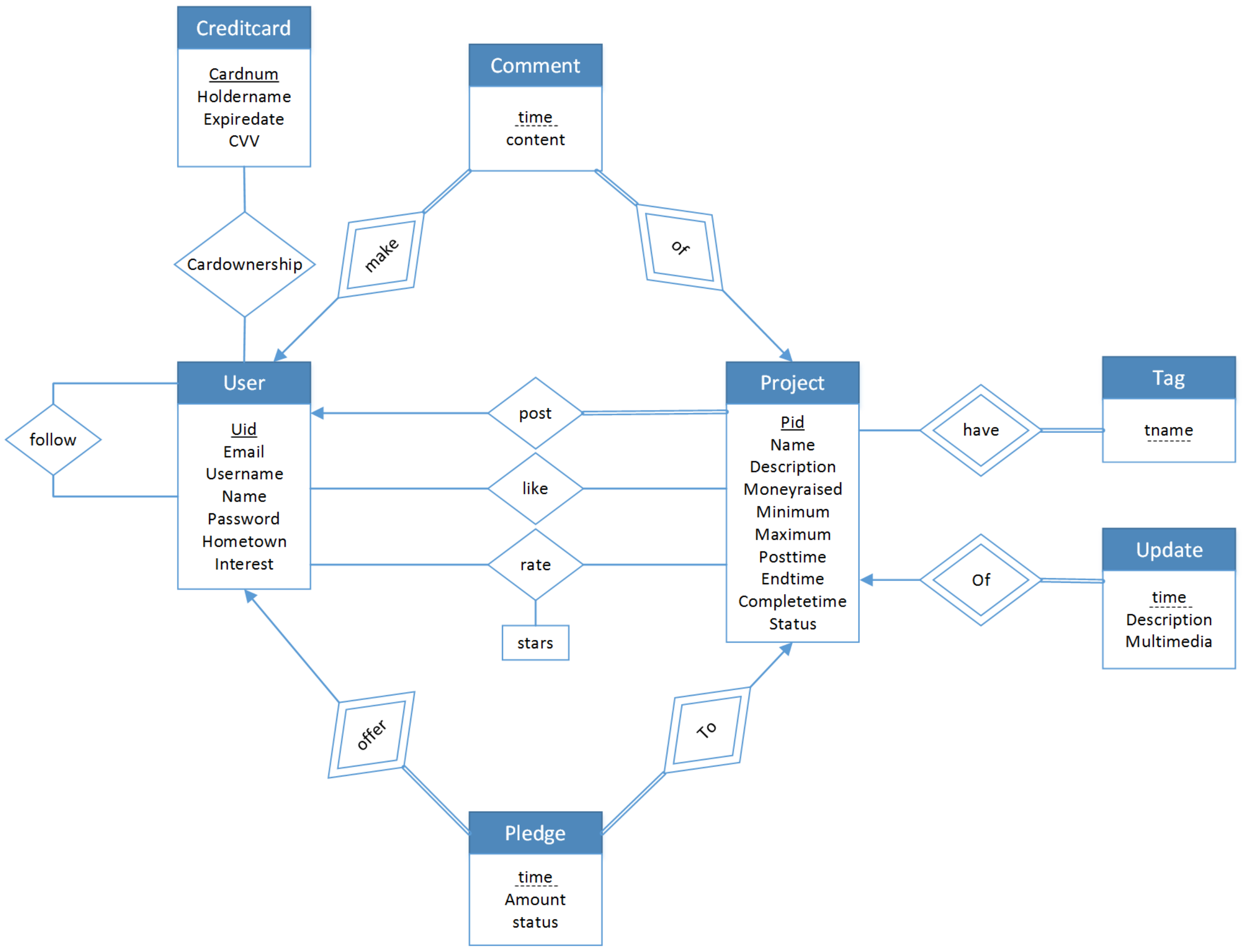
Computer Science and Engineering

CS6083, Spring 2017

zc950 Zexing Cheng

yz3330 Yuqi Zhang

ER diagram



Explanation of our design decisions:

In this design, the table USER is connected to table PROJECT with three relationships, which indicates that users can post their own projects, rate the projects they pledged and like the projects. Since a user can post more than one projects while a project can only belong to one owner, the relationship POST is a one-to-many relationship. And it is obviously that the relationships RATE and LIKE should be a many-to-many relationship. The attribute (status) in PROJECT can be 'ongoing', 'succeeded' and 'failed' to indicate the status of projects.

In reality, a user may own several credit cards and when he/she gives money to projects, he/she can choose a credit card from those he/she stored in system. Besides this, one credit card can be used by more than one person. For example, a son may use his father's credit card. So, it is efficient to use an extra table to store the information about credit cards, like card numbers, holder name, expired date and CVV, which are necessary to complete payment. And we use a relationship CARDOWNERSHIP to store the connection from USER to CREDITCARD. If we chose to store credit cards in USER, it would waste the space because we need to insert all information about the user whenever he/she adds a new card to his/her account.

The table COMMENT and PLEDGE are analogous in our design. They are all weak entities. Table COMMENT stores user's comments about projects. Table PLEDGE stores the amount of money given from users to projects. In this way, it is simple to manage that a user may give money to a project more than one time. Besides this, the attribute (status) stores the status of pledges. If the project is successfully funded, the status of pledges will become 'released', recording a charge is completed. Then the system will complete the charge, which doesn't need us to model. While if the project fails to be funded, the status will become 'refund', indicating the money will be given back to sponsors. Besides this, the status is ‘pending’ when the project is still raising money before it is successfully funded or the time is expired.

Table UPDATE is a weak entity used to store updates posted by owners of projects. It can store text or multimedia or both. Table TAG is a weak entity as well. At first, we use an attribute (tag) in table PROJECT to store tags or categories of the project. However, one project can have several tags. For example, a project about remixes of Justin Bieber’s songs is posted, then the tags can be ‘remix’, ‘pop’ and ‘Justin Bieber’. If we store tags in table PROJECT, then every new tag will lead to insert all information about the project. So, we use an extra table to store tags.

At last, we use a relationship FOLLOW to store the information that users may follow others.

Database Schema

USER (uid, email, username, name, password, hometown, interest)

CREDITCARD (cardnum, holdername, expiredate, CVV)

CARDOWNERSHIP (uid, cardnum)

PROJECT (pid, uid, name, description, moneyraised, minimum, maximum, posttime, endtime, completetime, status)

UPDATE (pid, time, description, multimedia)

FOLLOW (uid, followerID)

COMMENT (uid, pid, time, content)

RATE (uid, pid, stars)

LIKE (uid, pid)

TAG (pid, tname)

PLEDGE (uid, pid, time, amount, status)

Foreign key constraints:

(uid) in CARDOWNERSHIP references to (uid) in USER;

(cardnum) in CARDOWNERSHIP references to (cardnum) in CREDITCARD;

(uid) in PROJECT references to (uid) in USER;

(pid) in UPDATE references to (pid) in PROJECT;

(uid) in FOLLOW references to (uid) in USER;

(followerID) in FOLLOW references to (uid) in USER;

(uid) in COMMENT references to (uid) in USER;

(pid) in COMMENT references to (pid) in PROJECT;

(uid) in RATE references to (uid) in USER;

(pid) in RATE references to (pid) in PROJECT;

(uid) in LIKE references to (uid) in USER;

(pid) in LIKE references to (pid) in PROJECT;

(pid) in TAG reference to (pid) in PROJECT;

(uid) in PLEDGE reference to (uid) in USER;

(pid) in PLEDGE reference to (pid) in PROJECT;

Other constraints:

(status) in PROJECT: \`status\` ENUM('ongoing', 'succeeded', 'failed') DEFAULT “ongoing”;

(status) in PLEDGE: \`status\` ENUM('pending', 'released', 'refund') DEFAULT ‘pending'.

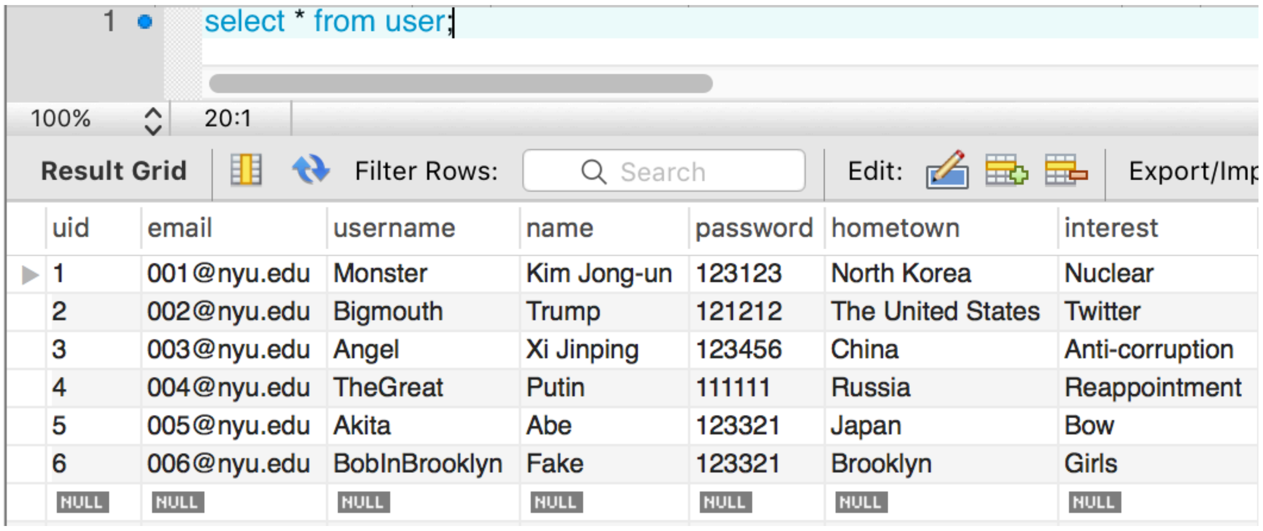
SQL queries:

- Create a record for a new user account, with a name, a login name, and a password.

/*In our design, user needs to provide email, name, and password when he/she creates the account. Their login names need to be set after they login in the system. So, email is equivalent to username here*/

INSERT INTO `user` (`name`, `email`, `password`) VALUES ('temp','22@nyu.edu', '111111');

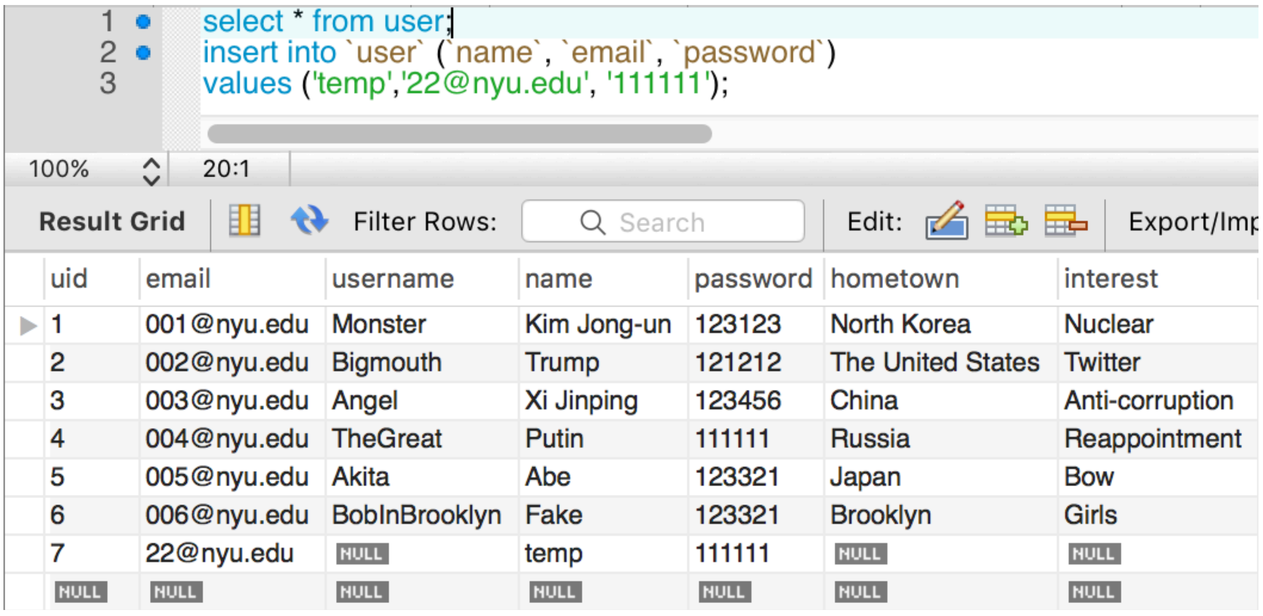
This is the USER table before insertion.



```
select * from user;
```

uid	email	username	name	password	hometown	interest
1	001@nyu.edu	Monster	Kim Jong-un	123123	North Korea	Nuclear
2	002@nyu.edu	Bigmouth	Trump	121212	The United States	Twitter
3	003@nyu.edu	Angel	Xi Jinping	123456	China	Anti-corruption
4	004@nyu.edu	TheGreat	Putin	111111	Russia	Reappointment
5	005@nyu.edu	Akita	Abe	123321	Japan	Bow
6	006@nyu.edu	BobInBrooklyn	Fake	123321	Brooklyn	Girls
NULL	NULL	NULL	NULL	NULL	NULL	NULL

This is the USER table after insertion.



```
select * from user;
insert into `user` (`name`, `email`, `password`)
values ('temp','22@nyu.edu', '111111');
```

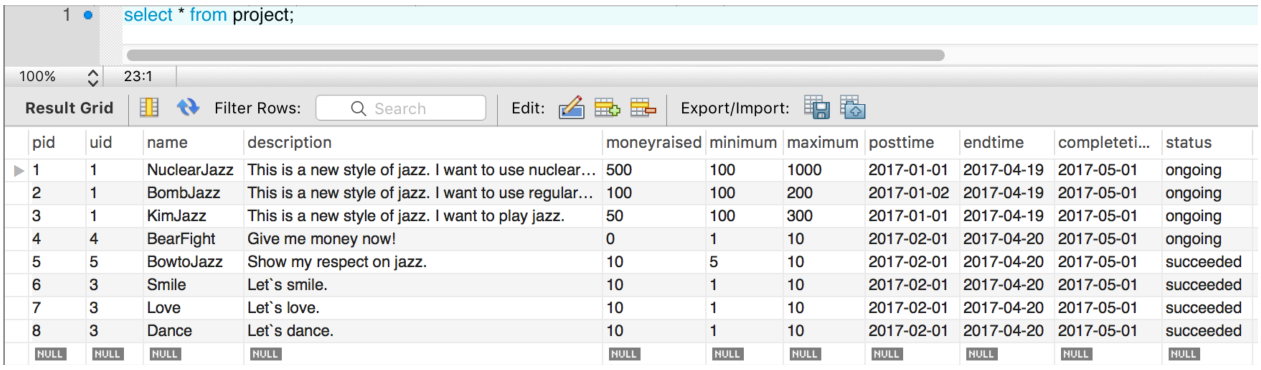
uid	email	username	name	password	hometown	interest
1	001@nyu.edu	Monster	Kim Jong-un	123123	North Korea	Nuclear
2	002@nyu.edu	Bigmouth	Trump	121212	The United States	Twitter
3	003@nyu.edu	Angel	Xi Jinping	123456	China	Anti-corruption
4	004@nyu.edu	TheGreat	Putin	111111	Russia	Reappointment
5	005@nyu.edu	Akita	Abe	123321	Japan	Bow
6	006@nyu.edu	BobInBrooklyn	Fake	123321	Brooklyn	Girls
7	22@nyu.edu	NULL	temp	111111	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

In our design, (uid) is auto-set by increment. The other attributes are set NULL as de-fault.

- List all projects that contain the keyword “jazz” and that are currently looking for funds, sorted in descending order by posting time.

```
SELECT pid, name, posttime
FROM project
WHERE description LIKE '%jazz%'
AND status = 'ongoing'
ORDER BY posttime DESC;
```

This is the PROJECT table.



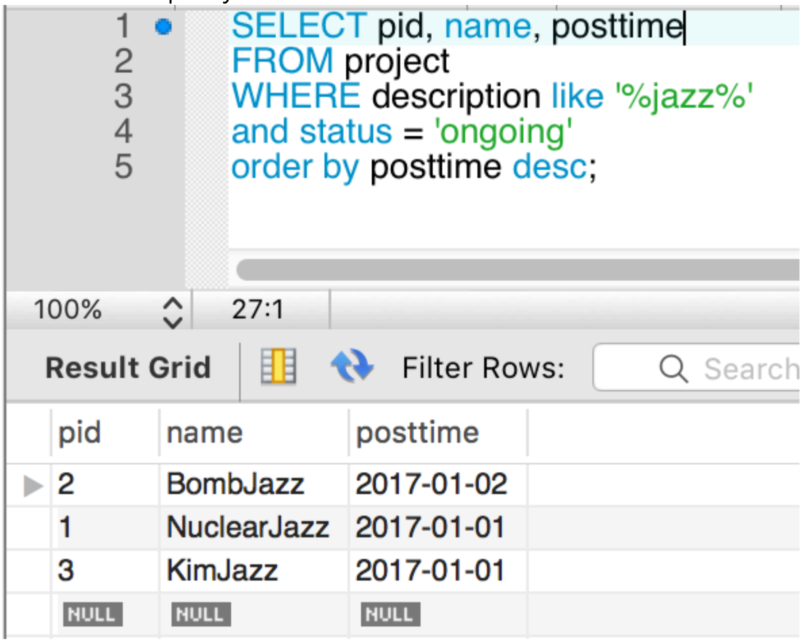
```
select * from project;
```

100% 23:1

Result Grid | Filter Rows: Search | Edit: | Export/Import:

pid	uid	name	description	moneyraised	minimum	maximum	posttime	endtime	completeti...	status
1	1	NuclearJazz	This is a new style of jazz. I want to use nuclear…	500	100	1000	2017-01-01	2017-04-19	2017-05-01	ongoing
2	1	BombJazz	This is a new style of jazz. I want to use regular…	100	100	200	2017-01-02	2017-04-19	2017-05-01	ongoing
3	1	KimJazz	This is a new style of jazz. I want to play jazz.	50	100	300	2017-01-01	2017-04-19	2017-05-01	ongoing
4	4	BearFight	Give me money now!	0	1	10	2017-02-01	2017-04-20	2017-05-01	ongoing
5	5	BowtoJazz	Show my respect on jazz.	10	5	10	2017-02-01	2017-04-20	2017-05-01	succeeded
6	3	Smile	Let`s smile.	10	1	10	2017-02-01	2017-04-20	2017-05-01	succeeded
7	3	Love	Let`s love.	10	1	10	2017-02-01	2017-04-20	2017-05-01	succeeded
8	3	Dance	Let`s dance.	10	1	10	2017-02-01	2017-04-20	2017-05-01	succeeded
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

This is the query result.



```
SELECT pid, name, posttime
FROM project
WHERE description like '%jazz%'
and status = 'ongoing'
order by posttime desc;
```

100% 27:1

Result Grid | Filter Rows: Search

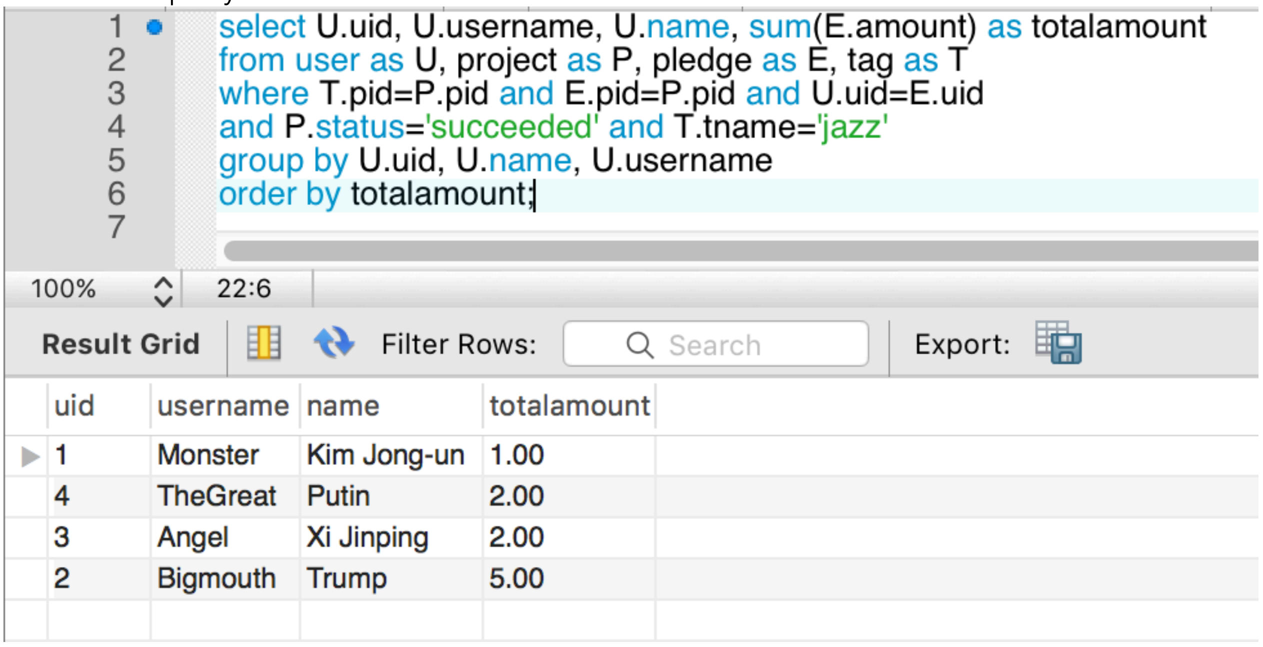
pid	name	posttime
2	BombJazz	2017-01-02
1	NuclearJazz	2017-01-01
3	KimJazz	2017-01-01
NULL	NULL	NULL

In PROJECT table, four rows contains keyword “jazz” and three of them are currently looking for funds, which are BombJazz, NuclearJazz and KimJazz. The query shows the correct result.

- List all users who have given money for projects containing the tag or category “jazz” in the past, sorted by the total amount they have successfully pledged (meaning, money that was actually charged).

```
SELECT U.uid, U.username, U.name, SUM(E.amount) AS totalamount
FROM user AS U, project AS P, pledge AS E, tag AS T
WHERE T.pid=P.pid AND E.pid=P.pid AND U.uid=E.uid
AND P.status='succeeded' AND T.tname='jazz'
GROUP BY U.uid
ORDER BY totalamount;
```

In this database, there is only one project whose tag is “jazz” and is funded successfully. This project is “BowtoJazz”, belonging to user “Abe”. Four users pledged this project, whose userIDs are 1, 2, 3 and 4.
This is the query result.



```
select U.uid, U.username, U.name, sum(E.amount) as totalamount
from user as U, project as P, pledge as E, tag as T
where T.pid=P.pid and E.pid=P.pid and U.uid=E.uid
and P.status='succeeded' and T.tname='jazz'
group by U.uid, U.name, U.username
order by totalamount;
```

100% 22:6

Result Grid Filter Rows: Search Export:

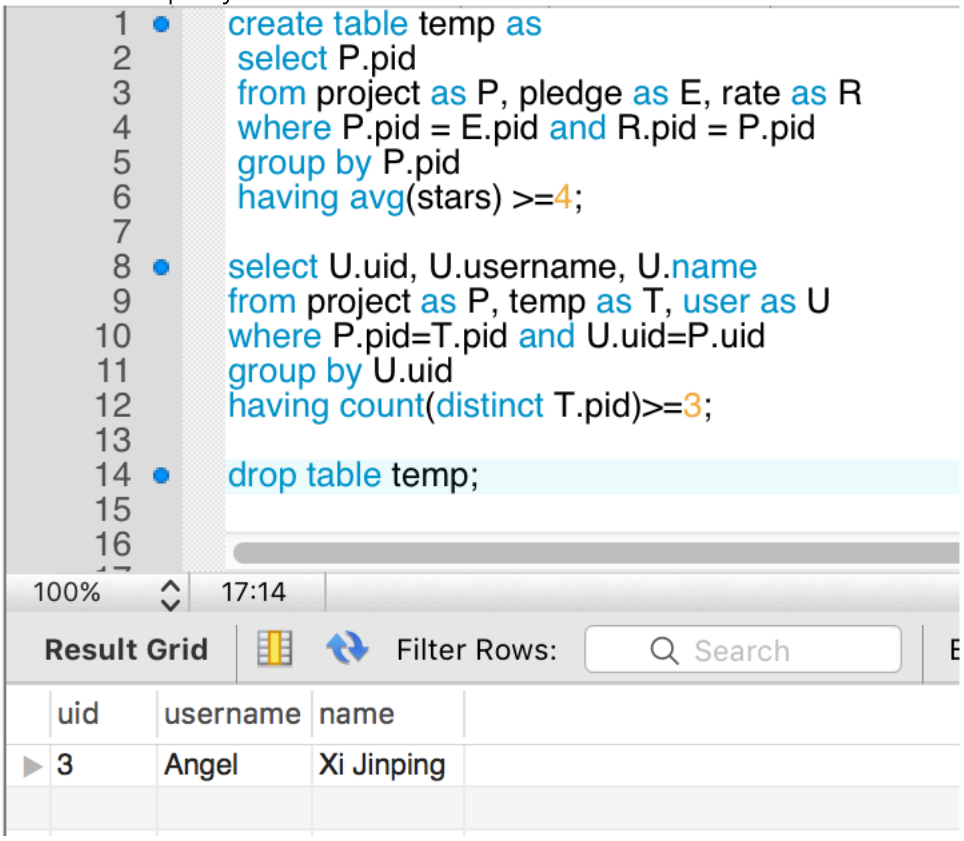
uid	username	name	totalamount
1	Monster	Kim Jong-un	1.00
4	TheGreat	Putin	2.00
3	Angel	Xi Jinping	2.00
2	Bigmouth	Trump	5.00

- List all users who have completed at least 3 projects, and where each of their projects received an average rating of 4 stars or higher from its sponsors.

```
CREATE TABLE temp AS
 SELECT P.pid
 FROM project AS P, pledge AS E, rate AS R
 WHERE P.pid = E.pid AND R.pid = P.pid
 GROUP BY P.pid
 HAVING AVG(stars) >=4;
SELECT U.uid, U.username, U.name
FROM project AS P, temp AS T, user AS U
WHERE P.pid=T.pid AND U.uid=P.uid
```

GROUP BY U.uid
HAVING COUNT(DISTINCT T.pid)>=3;
DROP TABLE temp;

In this database, only one user has completed three projects, whose username is “Angel”. His projects all get an average rating of 4 stars or higher.
This is the query result.



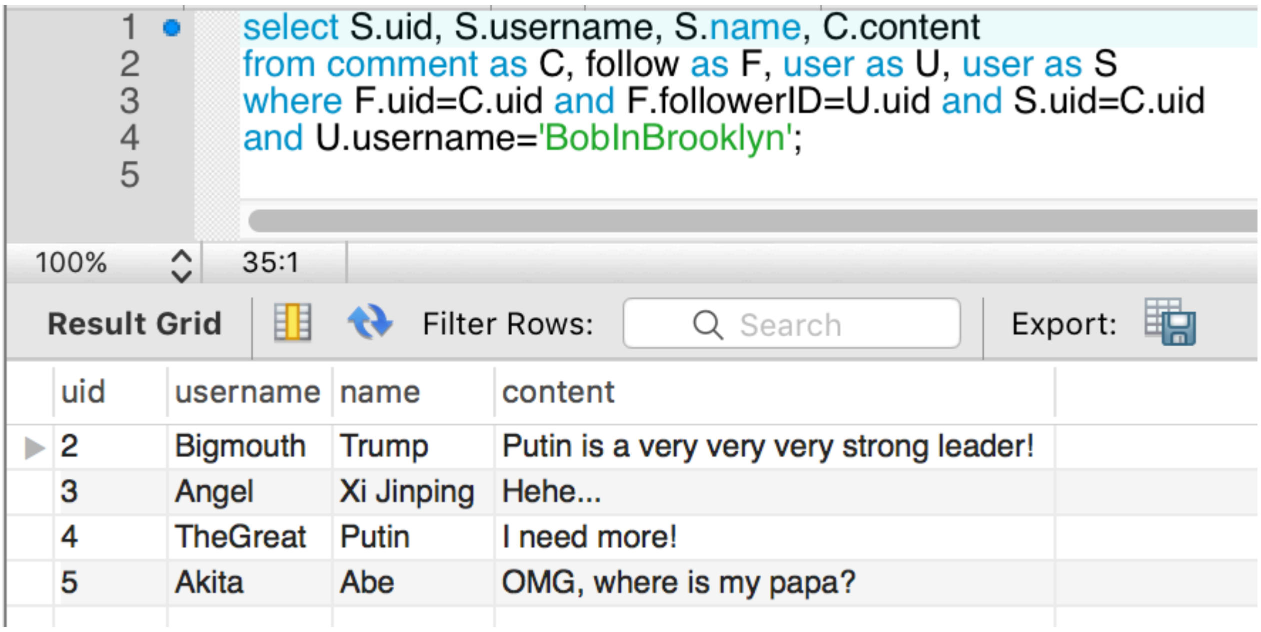
```
create table temp as
 select P.pid
 from project as P, pledge as E, rate as R
 where P.pid = E.pid and R.pid = P.pid
 group by P.pid
 having avg(stars) >=4;

select U.uid, U.username, U.name
from project as P, temp as T, user as U
where P.pid=T.pid and U.uid=P.uid
group by U.uid
having count(distinct T.pid)>=3;

drop table temp;
```

uid	username	name
3	Angel	Xi Jinping

- List all comments by users that are followed by user “BobInBrooklyn”.

```
SELECT S.uid, S.username, S.name, C.content
FROM comment AS C, follow AS F, user AS U, user AS S
WHEREF.uid=C.uid AND F.followerID=U.uid AND S.uid=C.uid
AND U.username='BobInBrooklyn';
```

This is the query result.



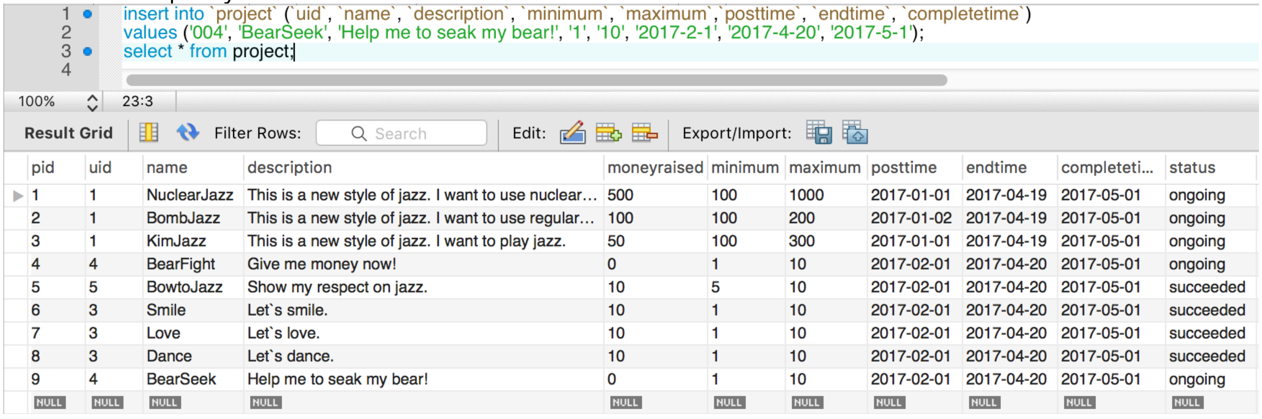
```
select S.uid, S.username, S.name, C.content
from comment as C, follow as F, user as U, user as S
where F.uid=C.uid and F.followerID=U.uid and S.uid=C.uid
and U.username='BobInBrooklyn';
```

uid	username	name	content
2	Bigmouth	Trump	Putin is a very very very strong leader!
3	Angel	Xi Jinping	Hehe...
4	TheGreat	Putin	I need more!
5	Akita	Abe	OMG, where is my papa?

- Insert a new project for a particular user, with a name, description, and other needed info.

INSERT INTO `project` (`uid`, `name`, `description`, `minimum`, `maximum`,`posttime`, `endtime`, `completetime`)
VALUES ('004', 'BearSeek', 'Help me to seek my bear!', '1', '10', '2017-2-1', '2017-4-20', '2017-5-1');

This is the query result.



```
insert into `project` (`uid`, `name`, `description`, `minimum`, `maximum`,`posttime`, `endtime`, `completetime`)
values ('004', 'BearSeek', 'Help me to seak my bear!', '1', '10', '2017-2-1', '2017-4-20', '2017-5-1');
select * from project;
```

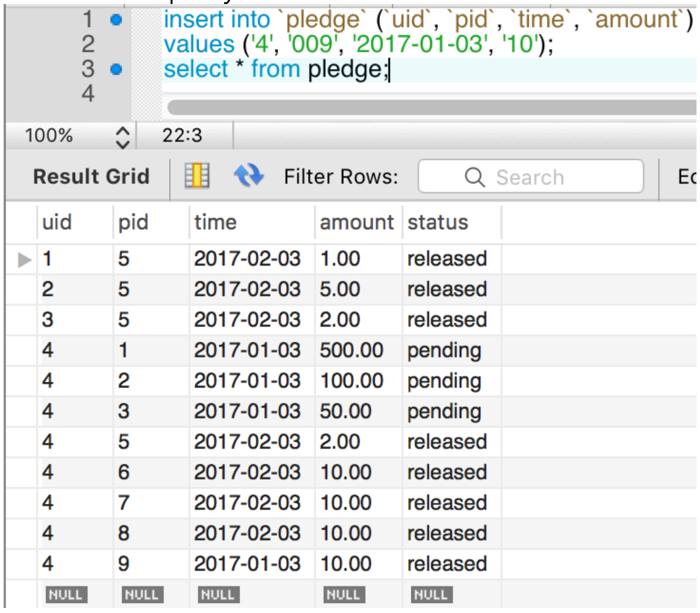
pid	uid	name	description	moneyraised	minimum	maximum	posttime	endtime	completeti...	status
1	1	NuclearJazz	This is a new style of jazz. I want to use nuclear...	500	100	1000	2017-01-01	2017-04-19	2017-05-01	ongoing
2	1	BombJazz	This is a new style of jazz. I want to use regular...	100	100	200	2017-01-02	2017-04-19	2017-05-01	ongoing
3	1	KimJazz	This is a new style of jazz. I want to play jazz.	50	100	300	2017-01-01	2017-04-19	2017-05-01	ongoing
4	4	BearFight	Give me money now!	0	1	10	2017-02-01	2017-04-20	2017-05-01	ongoing
5	5	BowtoJazz	Show my respect on jazz.	10	5	10	2017-02-01	2017-04-20	2017-05-01	succeeded
6	3	Smile	Let`s smile.	10	1	10	2017-02-01	2017-04-20	2017-05-01	succeeded
7	3	Love	Let`s love.	10	1	10	2017-02-01	2017-04-20	2017-05-01	succeeded
8	3	Dance	Let`s dance.	10	1	10	2017-02-01	2017-04-20	2017-05-01	succeeded
9	4	BearSeek	Help me to seak my bear!	0	1	10	2017-02-01	2017-04-20	2017-05-01	ongoing
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

It is clear that a new project is added into the table. (pid) is auto-set by increment and (moneyraised) is zero by default. (status) is "ongoing" by default.

- Insert a pledge to sponsor a project, for a particular user, project, and amount.

INSERT INTO `pledge` (`uid`, `pid`, `time`, `amount`)
VALUES ('4', '009', '2017-01-03', '10');

This is the query result.



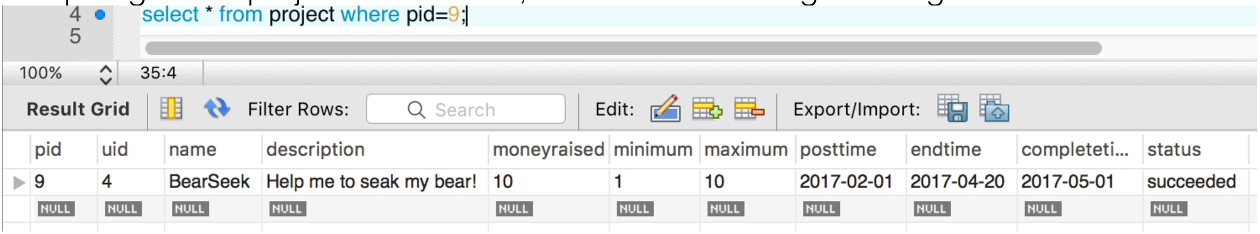
```
insert into `pledge` (`uid`, `pid`, `time`, `amount`)
values ('4', '009', '2017-01-03', '10');
select * from pledge;
```

100% 22:3

Result Grid Filter Rows: Search

uid	pid	time	amount	status
1	5	2017-02-03	1.00	released
2	5	2017-02-03	5.00	released
3	5	2017-02-03	2.00	released
4	1	2017-01-03	500.00	pending
4	2	2017-01-03	100.00	pending
4	3	2017-01-03	50.00	pending
4	5	2017-02-03	2.00	released
4	6	2017-02-03	10.00	released
4	7	2017-02-03	10.00	released
4	8	2017-02-03	10.00	released
4	9	2017-01-03	10.00	released
NULL	NULL	NULL	NULL	NULL

It is clear that a new record of pledge is added at the bottom of the table. Since someone pledged the project "BearSeek", some attributes might change.



```
select * from project where pid=9;
```

100% 35:4

Result Grid Filter Rows: Search Edit: Export/Import:

pid	uid	name	description	moneyraised	minimum	maximum	posttime	endtime	completeti...	status
9	4	BearSeek	Help me to seak my bear!	10	1	10	2017-02-01	2017-04-20	2017-05-01	succeeded
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

From this result, we can see that this project has be funded successfully, since one user just gave the maximum money of this project. So, the status tends to be "succeeded". And this is why the status in table PLEDGE is "released", indicating this charge is completed.

- Write queries for the end of a funding campaign. E.g., you could use triggers to detect when a campaign is fully funded or time is up; if successfully funded, generate charges to sponsors' credit cards.

Trigger 1: this trigger is used to update (moneyraised) in table PROJECT when someone gives money to this project.

```
DELIMITER I
CREATE TRIGGER update_moneyraised AFTER INSERT ON Pledge
FOR EACH ROW
```

```
BEGIN
        UPDATE Project
        SET moneyraised = moneyraised + NEW.amount
        WHERE pid = NEW.pid;
END |
DELIMITER ;
```

Trigger 2: this trigger is used to update (status) in table PLEDGE to reflect the changes of the status of project.

```
DELIMITER |
CREATE TRIGGER update_charge
AFTER update
  ON Project FOR EACH ROW
BEGIN
DECLARE sta VARCHAR(45);
SELECT status INTO sta FROM project WHERE pid=New.pid;
IF sta='succeeded' THEN
UPDATE pledge SET status='released' WHERE pid=NEW.pid;
END IF;
IF sta='failed' THEN
UPDATE pledge SET status='refunded' WHERE pid=NEW.PID;
END IF;
END |
DELIMITER ;
```

Procedure & Event: this event checks the money projects have already raised. Once the money meet the maximum amount or the end time has been expired, the status of project will be updated.

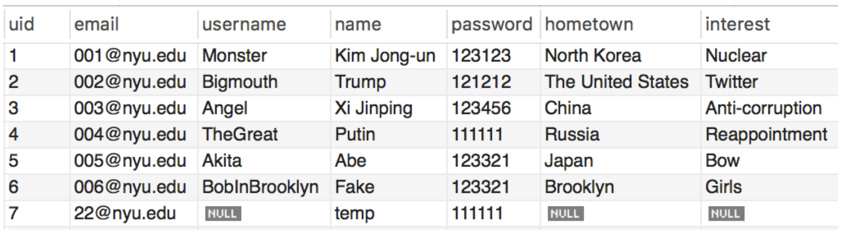
```
DELIMITER //
CREATE PROCEDURE update_status()
BEGIN
        update Project set project.status = 'succeeded' where moneyraised >= maximum;
        update Project set project.status = 'failed' where endtime < curdate() and mon-
eyraised < minimum;
        update Project set project.status = 'succeeded' where endtime < curdate() and
moneyraised >= minimum;
END //
DELIMITER ;

DROP EVENT IF EXISTS update_project;
CREATE EVENT update_project
on schedule EVERY 1 second
on completion preserve
do CALL update_status();
```

In our design, when user gives money to the project, money will be stored in a temporary place, like the account of the company of this webpage. When the project is still looking for money, the status of the project is “ongoing” and the status of the pledge is “tending”. These indicate that the money is temporarily stored now. The charge is actu-

ally generated. If the project get funded successfully, all the money given to this project will be transferred to the account of the owner of this project. The status of pledge becomes “released”, meaning the money is given to the project finally. But, if the project doesn’t get enough money before the time is up, then the money given to this project will be given back to sponsors. The status of the pledge will become “refunded”, meaning this amount of money is given back to sponsors.
So, in our design, the status of table PLEDGE reflects the circumstance of the pledge.The result shown in prior query has proved that these triggers and events can realize that when a campaign is fully funded or time is up, the charges will be generated or declined.

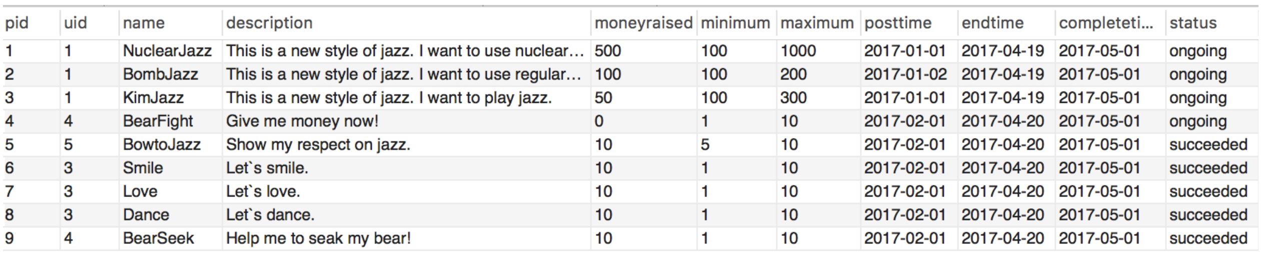
Sample data:

USER:



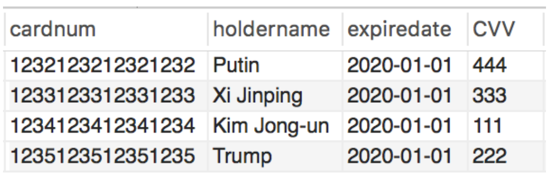
uid	email	username	name	password	hometown	interest
1	001@nyu.edu	Monster	Kim Jong-un	123123	North Korea	Nuclear
2	002@nyu.edu	Bigmouth	Trump	121212	The United States	Twitter
3	003@nyu.edu	Angel	Xi Jinping	123456	China	Anti-corruption
4	004@nyu.edu	TheGreat	Putin	111111	Russia	Reappointment
5	005@nyu.edu	Akita	Abe	123321	Japan	Bow
6	006@nyu.edu	BobInBrooklyn	Fake	123321	Brooklyn	Girls
7	22@nyu.edu	NULL	temp	111111	NULL	NULL

PROJECT:



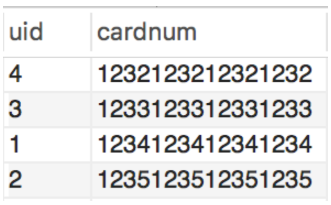
pid	uid	name	description	moneyraised	minimum	maximum	posttime	endtime	completeti…	status
1	1	NuclearJazz	This is a new style of jazz. I want to use nuclear…	500	100	1000	2017-01-01	2017-04-19	2017-05-01	ongoing
2	1	BombJazz	This is a new style of jazz. I want to use regular…	100	100	200	2017-01-02	2017-04-19	2017-05-01	ongoing
3	1	KimJazz	This is a new style of jazz. I want to play jazz.	50	100	300	2017-01-01	2017-04-19	2017-05-01	ongoing
4	4	BearFight	Give me money now!	0	1	10	2017-02-01	2017-04-20	2017-05-01	ongoing
5	5	BowtoJazz	Show my respect on jazz.	10	5	10	2017-02-01	2017-04-20	2017-05-01	succeeded
6	3	Smile	Let`s smile.	10	1	10	2017-02-01	2017-04-20	2017-05-01	succeeded
7	3	Love	Let`s love.	10	1	10	2017-02-01	2017-04-20	2017-05-01	succeeded
8	3	Dance	Let`s dance.	10	1	10	2017-02-01	2017-04-20	2017-05-01	succeeded
9	4	BearSeek	Help me to seak my bear!	10	1	10	2017-02-01	2017-04-20	2017-05-01	succeeded

CREDITCARD:



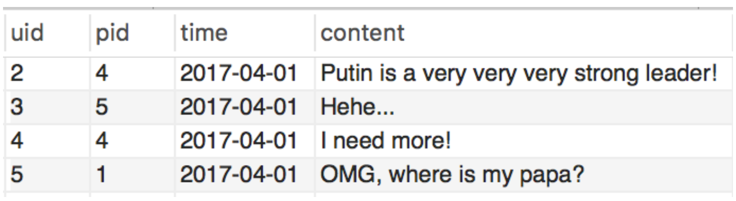
cardnum	holdername	expiredate	CVV
1232123212321232	Putin	2020-01-01	444
1233123312331233	Xi Jinping	2020-01-01	333
1234123412341234	Kim Jong-un	2020-01-01	111
1235123512351235	Trump	2020-01-01	222

CARDOWNERSHIP:



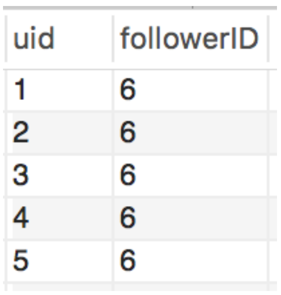
uid	cardnum
4	1232123212321232
3	1233123312331233
1	1234123412341234
2	1235123512351235

COMMENT:



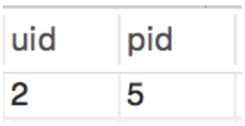
uid	pid	time	content
2	4	2017-04-01	Putin is a very very very strong leader!
3	5	2017-04-01	Hehe...
4	4	2017-04-01	I need more!
5	1	2017-04-01	OMG, where is my papa?

FOLLOW:



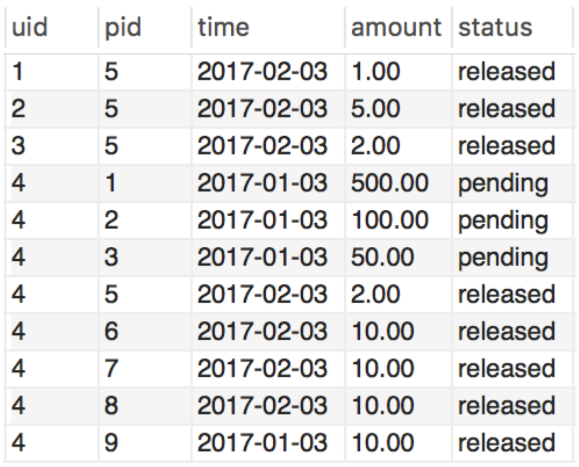
uid	followerID
1	6
2	6
3	6
4	6
5	6

LIKE:



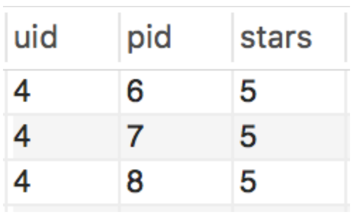
uid	pid
2	5

PLEDGE:



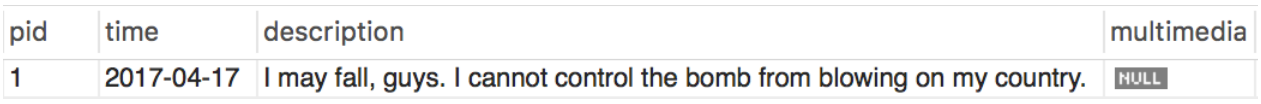
uid	pid	time	amount	status
1	5	2017-02-03	1.00	released
2	5	2017-02-03	5.00	released
3	5	2017-02-03	2.00	released
4	1	2017-01-03	500.00	pending
4	2	2017-01-03	100.00	pending
4	3	2017-01-03	50.00	pending
4	5	2017-02-03	2.00	released
4	6	2017-02-03	10.00	released
4	7	2017-02-03	10.00	released
4	8	2017-02-03	10.00	released
4	9	2017-01-03	10.00	released

RATE:



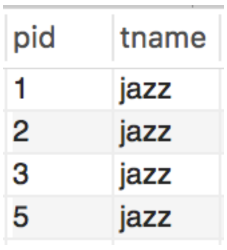
uid	pid	stars
4	6	5
4	7	5
4	8	5

UPDATE:



pid	time	description	multimedia
1	2017-04-17	I may fall, guys. I cannot control the bomb from blowing on my country.	NULL

TAG:



pid	tname
1	jazz
2	jazz
3	jazz
5	jazz