

FULL NAME : KHUSHI MUKESH SHAH

MySQL

GOODREADS DATABASE

To create and use database:

```
create database goodreads;  
use goodreads;
```

To create table author:

```
create table author(authorId int, name varchar(20) not null, primary key (authorId));
```

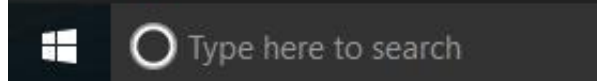
To insert values in author table:

```
insert into author(authorId, name) values(1, 'Ernest Hemingway');  
insert into author(authorId, name) values(2, 'Jane Austen'); insert  
into author(authorId, name) values(3, 'J.K. Rowling');
```

To show the values of author

table: select * from author;

```
mysql> select * from author;  
+-----+-----+  
| authorId | name          |  
+-----+-----+  
| 1        | Ernest Hemingway |  
| 2        | Jane Austen      |  
| 3        | J.K. Rowling     |  
+-----+-----+  
3 rows in set (0.04 sec)  
  
mysql> 
```



To create table book:

```
create table book (isbn varchar(255), title varchar(20), authorId int, numpages int not null, avgrating decimal(3,2), primary key(isbn), constraint fk1 foreign key(authorId) references author(authorId));
```

To insert values in author table:

```
insert into book(isbn, title, authorId, numpages, avgrating) values(1234567890, 'In our time', 1, 150, 4.75);
```

```
insert into book(isbn, title, authorId, numpages, avgrating) values(1234567980, 'Pride & Prejudice', 2, 350, 4.5);
```

```
insert into book(isbn, title, authorId, numpages, avgrating) values(1235674980, 'Harry Potter', 3, 550, 5.0);
```

```
insert into book(isbn, title, authorId, numpages, avgrating) values(1438265790, 'Emma', 2, 300, 4.75);
```

```
insert into book(isbn, title, authorId, numpages, avgrating) values(1586437290, 'The Silkworm', 3, 400, 5.0);
```

To show the values of book table:

```
select * from book;
```

```
mysql> select * from book;
```

isbn	title	authorId	numpages	avgrating
1234567890	In our time	1	150	4.75
1234567980	Pride & Prejudice	2	350	4.50
1235674980	Harry Potter	3	550	5.00
1438265790	Emma	2	300	4.75
1586437290	The Silkworm	3	400	5.00

```
5 rows in set (0.04 sec)
```

To create table users:

```
create table users(uid int, name varchar(20), age int, sex char(1), location varchar(20), birthday date, readCt int, toReadCt int, currentlyReadCt int, primary key(uid));
```

To insert values in users table:

```
insert into users(uid, name, age, sex, location, birthday, readCt, toReadCt, currentlyReadCt) values(1, 'Khushi Shah', 22, 'F', 'USA', '1995-10-03', 10, 5, 1);
```

```
insert into users(uid, name, age, sex, location, birthday, readCt, toReadCt, currentlyReadCt) values(2, 'Dhrumil Shah', 25, 'M', 'Australia', '1991-12-10', 8, 10, 2);
```

```
insert into users(uid, name, age, sex, location, birthday, readCt, toReadCt, currentlyReadCt) values(3, 'Khoobi Shah', 26, 'F', 'India', '1991-02-14', 10, 20, 5);
```

```
insert into users(uid, name, age, sex, location, birthday, readCt, toReadCt, currentlyReadCt) values(4, 'Sakhi Shah', 21, 'F', 'India', '1996-02-06', 8, 3, 7);
```

```
insert into users(uid, name, age, sex, location, birthday, readCt, toReadCt, currentlyReadCt) values(5, 'Spandan Shah', 28, 'M', 'UK', '1989-01-26', 10, 5, 8);
```

To show the values of users table:

```
select * from users;
```

```
mysql> select * from users;
```

uid	name	age	sex	location	birthday	readCt	toReadCt	currentlyReadCt
1	Khushi Shah	22	F	USA	1995-10-03	10	5	1
2	Dhrumil Shah	25	M	Australia	1991-12-10	8	10	2
3	Khoobi Shah	26	F	India	1991-02-14	10	20	5
4	Sakhi Shah	21	F	India	1996-02-06	8	3	7
5	Spandan Shah	28	M	UK	1989-01-26	10	5	8

```
5 rows in set (0.04 sec)
```

To create table shelf:

```
create table shelf(uid int, isbn varchar(255), name varchar(20), rating decimal(3,2), dateRead date,
dateAdded date, primary key(uid, isbn), constraint fk2 foreign key(uid) references users(uid),
constraint fk3 foreign key(isbn) references book(isbn));
```

To insert values in shelf table:

```
insert into shelf(uid, isbn, name, rating, dateRead, dateAdded) values(1, 1234567890, 'Read', 4.5,
'2010-01-01', '2010-03-01');
```

```
insert into shelf(uid, isbn, name, rating, dateRead, dateAdded) values(2, 1234567980, 'Read', 4.5,
'2010-02-01', '2010-06-01');
```

```
insert into shelf(uid, isbn, name, rating, dateRead, dateAdded) values(3, 1438265790, 'Read', 5.0,
'2010-08-01', '2010-12-01');
```

```
insert into shelf(uid, isbn, name, rating, dateRead, dateAdded) values(3, 1586437290, 'Currently-
Reading', 5.0, '2011-04-01', '2012-12-01');
```

```
insert into shelf(uid, isbn, name, rating, dateRead, dateAdded) values(4, 1234567890, 'To-Read', 5.0,
'2011-04-01', '2011-12-01');
```

```
insert into shelf(uid, isbn, name, rating, dateRead, dateAdded) values(4, 1438265790, 'To-Read', 4.5,
'2011-08-01', '2011-12-01');
```

```
insert into shelf(uid, isbn, name, rating, dateRead, dateAdded) values(4, 1586437290, 'Read', 5.0,
'2011-09-01', '2011-12-01');
```

```
insert into shelf(uid, isbn, name, rating, dateRead, dateAdded) values(5, 1235674980, 'Read', 5.0,
'2012-01-01', '2011-12-01');
```

To show the values of shelf table:

```
select * from shelf;
```

```
mysql> select * from shelf;
```

uid	isbn	name	rating	dateRead	dateAdded
1	1234567890	Read	4.50	2010-01-01	2010-03-01
2	1234567980	Read	4.50	2010-02-01	2010-06-01
3	1438265790	Read	5.00	2010-08-01	2010-12-01
3	1586437290	Currently-Reading	5.00	2011-04-01	2012-12-01
4	1234567890	To-Read	5.00	2011-04-01	2011-12-01
4	1438265790	To-Read	4.50	2011-08-01	2011-12-01
4	1586437290	Read	5.00	2011-09-01	2011-12-01
5	1235674980	Read	5.00	2012-01-01	2011-12-01

8 rows in set (0.15 sec)

To create table friends:

```
create table friends(uid int, fid int, primary key(uid, fid), constraint fk4 foreign key(uid) references users(uid), constraint fk5 foreign key(fid) references users(uid));
```

To insert values in friends table:

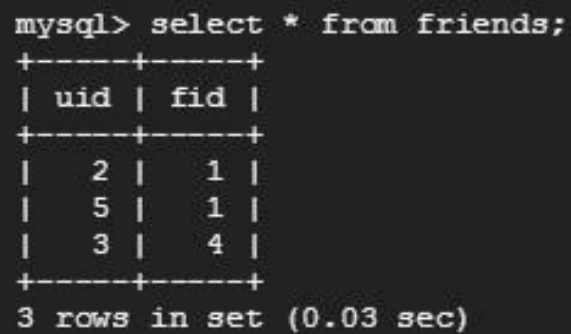
```
insert into friends(uid, fid) values(2, 1);
```

```
insert into friends(uid, fid) values(5, 1);
```

```
insert into friends(uid, fid) values(3, 4);
```

To show the values of friends table:

```
select * from friends;
```



```
mysql> select * from friends;
+-----+-----+
| uid | fid |
+-----+-----+
| 2   | 1   |
| 5   | 1   |
| 3   | 4   |
+-----+-----+
3 rows in set (0.03 sec)
```

Answer-1

User adds a new book to his shelf with a rating. Update the average rating of that book.

Here, first a new book is added by user in his shelf using insert query. So the query goes as follows:

```
insert into shelf(uid, isbn, name, rating, dateRead, dateAdded) values(1, 1235674980, 'Read', 4.0, '2017-01-01', '2017-10-01');
```

The output can be seen using:

```
select * from shelf;
```

UID 1 has now read 2 books, which is reflected in the table.

```
mysql> insert into shelf(uid, isbn, name, rating, dateRead, dateAdded) values(1, 1235674980, 'Read', 4.0, '2017-01-01', '2017-10-01');
Query OK, 1 row affected (0.04 sec)

mysql> select * from shelf;
```

uid	isbn	name	rating	dateRead	dateAdded
1	1234567890	Read	4.50	2010-01-01	2010-03-01
1	1235674980	Read	4.00	2017-01-01	2017-10-01
2	1234567980	Read	4.50	2010-02-01	2010-06-01
3	1438265790	Read	5.00	2010-08-01	2010-12-01
3	1586437290	Currently-Reading	5.00	2011-04-01	2012-12-01
4	1234567890	To-Read	5.00	2011-04-01	2011-12-01
4	1438265790	To-Read	4.50	2011-08-01	2011-12-01
4	1586437290	Read	5.00	2011-09-01	2011-12-01
5	1235674980	Read	5.00	2012-01-01	2011-12-01

```
9 rows in set (0.04 sec)
```

In the second part we update the average rating of the book, since one more rating is added for the book. First we need to find the average of rating of the book with particular isbn from shelf table and then update the average rating of that particular isbn in book table.

The query here goes as follows:

update book set avgrating=(select avg(rating) from shelf where isbn='1235674980') where isbn='1235674980';

```
mysql> update book set avgrating=(select avg(rating) from shelf where isbn='1235674980') where isbn='1235674980';
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from book;
```

isbn	title	authorId	numpages	avgrating
1234567890	In our time	1	150	4.75
1234567980	Pride & Prejudice	2	350	4.50
1235674980	Harry Potter	3	550	4.50
1438265790	Emma	2	300	4.75
1586437290	The Silkworm	3	400	5.00

```
5 rows in set (0.03 sec)
```

So we can see here that the average rating of Harry Potter has dropped from 5.00 to 4.50.

Answer-2

Find the names of the common books that were read by any two users X and Y.

We use self join here to compare the isbn in the same table for given two userids. So after getting that particular isbn or list of isbn (common between two users), we select the title of that isbn from book table to show the result. We use IN because there is a possibility that multiple isbns can be common between two users.

The query goes as follows:

select title from book where isbn in (select s.isbn from shelf s, shelf s1 where s.isbn=s1.isbn and s.uid=3 and s1.uid=4);

```
mysql> select title from book where isbn in (select s.isbn from shelf s,shelf s1 where s.isbn=s1.isbn and s.uid=3 and s1.uid=4);
+-----+
| title      |
+-----+
| Emma       |
| The Silkworm |
+-----+
2 rows in set (0.04 sec)
```

```
mysql> select title from book where isbn in (select s.isbn from shelf s,shelf s1 where s.isbn=s1.isbn and s.uid=5 and s1.uid=1);
+-----+
| title      |
+-----+
| Harry Potter |
+-----+
1 row in set (0.12 sec)
```

As a result, the common books read between uid 3 and 4 are Emma, The Silkworm.

The common book read between uid 1 and 5 is Harry Potter.

GITHUB

To create and use database:

```
create database github;
use github;
```

```
mysql> create database github;
Query OK, 1 row affected (0.04 sec)

mysql> use github;
Database changed
```

CREATE TABLES

```
create table users (userId int, noOfRepos int, location varchar(50), email varchar(50), website varchar(50),
contributions int, primary key(userId));
```

```
create table repository (repoId int, userId int not null, issueCount int, pullCount int, projectsCount int, wiki
boolean primary key (repoId), constraint fk1 foreign key(userId) references users(userId));
```

```
create table issue (issueId int, creatorId int not null, raiseDate date, resolverId int, resolveDate date, primary key
(issueId), constraint fk2 foreign key(creatorId) references users(userId), constraint fk3 foreign key(resolverId)
references users(userId));
```

```
create table codes (repoId int, commits int not null, branches int not null, releases int, contributors int, primary
key(repoId), constraint fk4 foreign key(repoId) references repository(repoId));
```

```
create table branch (branchId int, repoId int not null, userId int not null, primary key(branchId), constraint fk5 foreign
key(repoId) references repository(repoId), constraint fk6 foreign key(userId) references users(userId));
```

```
create table commits (commitId int, branchId int not null, commitTime datetime, noOfFiles int, additions int, deletions
int, primary key (commitId), constraint fk7 foreign key(branchId) references branch(branchId));
```

```
mysql> create table users(userId int, noOfRepos int, location varchar(50), email varchar(50), website varchar(50), contributions int, primary key(userId));
Query OK, 0 rows affected (0.06 sec)

mysql> create table repository(repoId int, userId int not null, issueCount int, pullCount int, projectsCount int, wiki boolean, primary key(repoId), constraint fk1 foreign key
(userId) references users(userId));
Query OK, 0 rows affected (0.06 sec)

mysql> create table issue (issueId int, creatorId int not null, raiseDate date, resolverId int, resolveDate date, primary key(issueId), constraint fk2 foreign key(creatorId)
references users(userId), constraint fk3 foreign key(resolverId) references users(userId));
Query OK, 0 rows affected (0.06 sec)

mysql> create table codes(repoId int, commits int not null, branches int not null, releases int, contributors int, primary key(repoId), constraint fk4 foreign key(repoId) re
ferences repository(repoId));
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> create database github;
Query OK, 1 row affected (0.04 sec)

mysql> use github;
Database changed
mysql> create table users(userId int, noOfRepos int, location varchar(50), email varchar(50), website varchar(50), contributions int, primary key(userId));
Query OK, 0 rows affected (0.06 sec)

mysql> create table repository(repoId int, userId int not null, issueCount int, pullCount int, projectsCount int, wiki boolean, primary key(repoId), constraint fk1 foreign key (userId) references users(userId));
Query OK, 0 rows affected (0.06 sec)

mysql> create table issue (issueId int, creatorId int not null, raiseDate date, resolverId int, resolveDate date, primary key(issueId), constraint fk2 foreign key(creatorId) references users(userId), constraint fk3 foreign key (resolverId) references users(userId));
Query OK, 0 rows affected (0.06 sec)

mysql> create table codes(repoId int, commits int not null, branches int not null, releases int, contributors int, primary key(repoId), constraint fk4 foreign key(repoId) references repository(repoId));
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> create table branch (branchId int, repoId int not null, userId int not null, primary key(branchId), constraint fk5 foreign key(repoId) -> references repository(repoId), constraint fk6 foreign key(userId) references users(userId));
Query OK, 0 rows affected (0.07 sec)

mysql> create table commits (commitId int, branchId int not null, commitTime datetime, noOfFiles int, additions int, deletions int, primary key -> (commitId), constraint fk7 foreign key(branchId) references branch(branchId));
Query OK, 0 rows affected (0.06 sec)
```

INSERT VALUES IN TABLES

insert into users(userId, noOfRepos, location, email, website, contribution) values(1, 15, 'jampot', 'khushish@usc.edu', 'abc.com', 100);

insert into users(userId, noOfRepos, location, email, website, contribution) values(2, 20, 'jampot', 'shailypa@usc.edu', 'xyz.com', 240);

insert into repository(repoId, userId, issueCount, pullCount, projectsCount, wiki) values(1,1,10,10,10,1);

insert into repository(repoId, userId, issueCount, pullCount, projectsCount, wiki) values(2,1,10,10,10,1);

insert into repository(repoId, userId, issueCount, pullCount, projectsCount, wiki)

```
values(3,2,10,10,10,1);
```

```
mysql> insert into repository(repoId, userId, issueCount, pullCount, projectsCount, wiki) values(3,2,10,10,10,1);
Query OK, 1 row affected (0.05 sec)

mysql> select * from repository;
+-----+-----+-----+-----+-----+-----+
| repoId | userId | issueCount | pullCount | projectsCount | wiki |
+-----+-----+-----+-----+-----+-----+
| 1 | 1 | 10 | 10 | 10 | 1 |
| 2 | 1 | 10 | 10 | 10 | 1 |
| 3 | 2 | 10 | 10 | 10 | 1 |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.03 sec)
```

```
mysql> insert into users(userId, noOfRepos, location, email, website, contributions) values(1, 15, 'jampot', 'khushish@usc.edu', 'abc.com', 100);
Query OK, 1 row affected (0.05 sec)

mysql> insert into users(userId, noOfRepos, location, email, website, contributions) values(2, 20, 'jampot', 'shailypa@usc.edu', 'xyz.com', 240);
Query OK, 1 row affected (0.04 sec)

mysql> select * from users;
+-----+-----+-----+-----+-----+-----+
| userId | noOfRepos | location | email | website | contributions |
+-----+-----+-----+-----+-----+-----+
| 1 | 15 | jampot | khushish@usc.edu | abc.com | 100 |
| 2 | 20 | jampot | shailypa@usc.edu | xyz.com | 240 |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.04 sec)

mysql> insert into repository(repoId, userId, issueCount, pullCount, projectsCount, wiki) values(1, 1, 10, 10, 10, 1);
Query OK, 1 row affected (0.04 sec)

mysql> insert into repository(repoId, userId, issueCount, pullCount, projectsCount, wiki) values(2, 1, 10, 10, 10, 1);
Query OK, 1 row affected (0.05 sec)

mysql> select * from repository;
+-----+-----+-----+-----+-----+-----+
| repoId | userId | issueCount | pullCount | projectsCount | wiki |
+-----+-----+-----+-----+-----+-----+
| 1 | 1 | 10 | 10 | 10 | 1 |
| 2 | 1 | 10 | 10 | 10 | 1 |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.04 sec)
```

```
mysql> insert into users(userId, noOfRepos, location, email, website, contributions) values(3,25,'jampot','virali@gmail.com','lmn.com',360);
Query OK, 1 row affected (0.04 sec)

mysql> insert into users(userId, noOfRepos, location, email, website, contributions) values(4,35,'jampot','sakhi@gmail.com','pqr.com',360);
Query OK, 1 row affected (0.04 sec)

mysql> select * from users;
+-----+-----+-----+-----+-----+-----+
| userId | noOfRepos | location | email | website | contributions |
+-----+-----+-----+-----+-----+-----+
| 1 | 15 | jampot | khushish@usc.edu | abc.com | 100 |
| 2 | 20 | jampot | shailypa@usc.edu | xyz.com | 240 |
| 3 | 25 | jampot | virali@gmail.com | lmn.com | 360 |
| 4 | 35 | jampot | sakhi@gmail.com | pqr.com | 360 |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.04 sec)
```

insert into users(userId, noOfRepos, location, email, website, contribution) values(3, 25, 'jampot', 'virali@gmail.com', 'lmn.com', 360);

insert into users(userId, noOfRepos, location, email, website, contribution) values(4, 35, 'jampot', 'sakhi@gmail.com', 'pqr.com', 360);

insert into issue(issueId, creatorId, raiseDate, resolverId, resolveDate) values(1,1,'2017-01-01',2,'2017-02-02');

insert into issue(issueId, creatorId, raiseDate, resolverId, resolveDate) values(2,1,'2017-01-01',2,'2017-02-02');

```
insert into issue(issueId, creatorId, raiseDate, resolverId, resolveDate) values(3,2,'2017-01-01',2,'2017-02-02');
```

```
insert into issue(issueId, creatorId, raiseDate, resolverId, resolveDate) values(4,2,'2017-01-01',1,'2017-02-02');
```

```
mysql> insert into issue(issueId, creatorId, raiseDate, resolverId, resolveDate) values(2, 1, '2017-01-01', 2, '2017-02-02');
Query OK, 1 row affected (0.04 sec)

mysql> insert into issue(issueId, creatorId, raiseDate, resolverId, resolveDate) values(3, 2, '2017-01-01', 2, '2017-02-02');
Query OK, 1 row affected (0.05 sec)

mysql> insert into issue(issueId, creatorId, raiseDate, resolverId, resolveDate) values(4, 2, '2017-01-01', 1, '2017-02-02');
Query OK, 1 row affected (0.05 sec)

mysql> select * from issue;
+-----+-----+-----+-----+-----+
| issueId | creatorId | raiseDate | resolverId | resolveDate |
+-----+-----+-----+-----+-----+
| 1 | 1 | 2017-01-01 | 2 | 2017-02-02 |
| 2 | 1 | 2017-01-01 | 2 | 2017-02-02 |
| 3 | 2 | 2017-01-01 | 2 | 2017-02-02 |
| 4 | 2 | 2017-01-01 | 1 | 2017-02-02 |
+-----+-----+-----+-----+-----+
4 rows in set (0.04 sec)
```

insert into codes(repoId, commits, branches, releases, contributors) values(1,2,1,1,2); insert
into codes(repoId, commits, branches, releases, contributors) values(2,2,1,1,2);

```
mysql> insert into codes(repoId, commits, branches, releases, contributors) values(1,2,1,1,2);
Query OK, 1 row affected (0.04 sec)

mysql> insert into codes(repoId, commits, branches, releases, contributors) values(2,2,1,1,2);
Query OK, 1 row affected (0.04 sec)

mysql> select * from codes;
+-----+-----+-----+-----+-----+
| repoId | commits | branches | releases | contributors |
+-----+-----+-----+-----+-----+
| 1 | 2 | 1 | 1 | 2 |
| 2 | 2 | 1 | 1 | 2 |
+-----+-----+-----+-----+-----+
2 rows in set (0.04 sec)
```

insert into branch(branchId, repoId, userId) values(1,1,1); insert
into branch(branchId, repoId, userId) values(2,2,2);

```
mysql> insert into branch(branchId, repoId, userId) values(2,2,2);
Query OK, 1 row affected (0.04 sec)

mysql> select * from branch;
+-----+-----+-----+
| branchId | repoId | userId |
+-----+-----+-----+
| 1 | 1 | 1 |
| 2 | 2 | 2 |
+-----+-----+-----+
2 rows in set (0.04 sec)
```

insert into branch(branchId, repold, userId) values(3,2,2);

```
mysql> select * from branch;
+-----+-----+-----+
| branchId | repold | userId |
+-----+-----+-----+
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 3 | 2 | 2 |
+-----+-----+-----+
3 rows in set (0.03 sec)
```

insert into commits(commitId, branchId, commitTime, noOfFiles, additions, deletions)
values(1,1,'2017-01-01 11:00:00',2,1000,2000);

insert into commits(commitId, branchId, commitTime, noOfFiles, additions, deletions)
values(2,1,'2017-01-01 11:00:00',2,100,20000);

```
mysql> insert into commits(commitId, branchId, commitTime, noOfFiles, additions, deletions) values(1,1,'2017-01-01 11:00:00',2,1000,2000);
Query OK, 1 row affected (0.05 sec)

mysql> insert into commits(commitId, branchId, commitTime, noOfFiles, additions, deletions) values(2,1,'2017-01-01 11:00:00',2,100,20000);
Query OK, 1 row affected (0.04 sec)

mysql> select * from commits;
+-----+-----+-----+-----+-----+-----+
| commitId | branchId | commitTime | noOfFiles | additions | deletions |
+-----+-----+-----+-----+-----+-----+
| 1 | 1 | 2017-01-01 11:00:00 | 2 | 1000 | 2000 |
| 2 | 1 | 2017-01-01 11:00:00 | 2 | 100 | 20000 |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.05 sec)
```

Answer-1

Find the users who made branches of either of repositories X or Y but not of a repository Z.

Here first we find the branchId whose repository Id can be x or y but not z. Then we find the user ids whose branch id are the result of previous statement. Using 'in' keyword helps to fetch and equate multiple values instead of single value from the row/subquery. We use '!=' for not equal to.

Here we are checking for the repold which are 1 or 2 but not 3. Then based on its branch id we fetch the user id.

The query goes as follows:

select userId from branch where branchId in (select branchId from branch where repold in (1,2) and repold!=3);

```
mysql> select userId from branch where branchId in (select branchId from branch where repold in (1,2) and repold!=3);
+-----+
| userId |
+-----+
|      1 |
|      2 |
|      2 |
+-----+
3 rows in set (0.05 sec)

mysql> select userId from branch where branchId in (select branchId from branch where repold in (1,3) and repold!=2);
+-----+
| userId |
+-----+
|      1 |
+-----+
1 row in set (0.04 sec)
```

```
mysql> select userId from branch where branchId in (select branchId from branch where repold in (3,2) and repold!=1);
+-----+
| userId |
+-----+
|      2 |
|      2 |
+-----+
2 rows in set (0.03 sec)

mysql> select userId from branch where branchId in (select branchId from branch where repold in (3,1) and repold!=2);
+-----+
| userId |
+-----+
|      1 |
+-----+
1 row in set (0.04 sec)

mysql> select userId from branch where branchId in (select branchId from branch where repold in (2,1) and repold!=3);
+-----+
| userId |
+-----+
|      1 |
|      2 |
|      2 |
+-----+
3 rows in set (0.04 sec)
```


Answer-2

Find the top commit with the highest lines of code reduced. (Hint: We need to find the maximized value of: number of deletions - number of additions in each commit).

To find the top commit with highest lines of code reduced, we first find the maximum of (deletions-additions) and then select its corresponding commitId.

The query goes as follows:

select commitId from commits where (deletions-additions) = (select max(deletions-additions) from commits);

```
mysql> select commitId from commits where (deletions-additions) = (select max(deletions-additions) from commits);
+-----+
| commitId |
+-----+
|          2 |
+-----+
1 row in set (0.03 sec)
```

Answer-3

(BONUS question) List the users who solved more issues than they raised. (i.e. number of issues in which they were the resolver is greater than the number of issues where they were the creator.)

Here I am using the previously created table and its data entries. For a condition where there can be a user who has resolved an issue but not created one, I added one extra entry using:

insert into issue(issueId, creatorId, raiseDate, resolverId, resolveDate);

And after checking with the above test case, we will go with the case where there are issues created as well as resolved by a user where the number of issues resolved by the user are more (not null values.) Here we will delete the previously inserted entry using:

delete from issue where resolverId=3;

In this type, there can be users who have never raised an issue but resolved one or more, or there can be users who have resolved more issues then created. We group by creator id and resolver id and count the same for each user. (resolver id and creator id are nothing but user ids statinf that that particular user is a creator or resolver of that particular issue.)

We use right join between creator and user because in case if there is a user who has created more issues than resolved, or a user that has never resolved any issues but only created, can return null value and that won't make any difference to our query since we are not looking for users whose count of resolving issues is less than count of creating issues. So in case of null values of resolving issues as well, the query won't give incorrect result.

The where clause has a condition for the case where the user has resolved one or more issues and created none, as well as it has condition where the user has resolved more issues than created.

The query here goes as follows:

**select B.resolverId as Id from (select creatorId, count(*) as y1 from issue group by creatorId) A
right join (select resolverId, count(*) as y2 from issue group by resolverId) B on
A.creatorId=B.resolverId where ((A.y1<B.y2) or (A.y1 IS NULL and B.y2 is not NULL));**

```
mysql> select * from issue;
+-----+-----+-----+-----+
| issueId | creatorId | raiseDate | resolverId | resolveDate |
+-----+-----+-----+-----+
| 1 | 1 | 2017-01-01 | 2 | 2017-02-02 |
| 2 | 1 | 2017-01-01 | 2 | 2017-02-02 |
| 3 | 2 | 2017-01-01 | 2 | 2017-02-02 |
| 4 | 2 | 2017-01-01 | 1 | 2017-02-02 |
| 5 | 2 | 2017-03-01 | 3 | 2017-04-04 |
+-----+-----+-----+-----+
5 rows in set (0.04 sec)

mysql> select B.resolverId as Id from (select creatorId, count(*) as y1 from issue group by creatorId) A right join (select resolverId, count(*) as y2 from issue group by r
esolverId) B on A.creatorId=B.resolverId where ((A.y1<B.y2) or (A.y1 IS NULL and B.y2 is not NULL));
+-----+
| Id |
+-----+
| 3 |
+-----+
1 row in set (0.04 sec)
```

```
mysql> delete from issue where resolverId=3;
Query OK, 1 row affected (0.05 sec)

mysql> select * from issue;
+-----+-----+-----+-----+
| issueId | creatorId | raiseDate | resolverId | resolveDate |
+-----+-----+-----+-----+
| 1 | 1 | 2017-01-01 | 2 | 2017-02-02 |
| 2 | 1 | 2017-01-01 | 2 | 2017-02-02 |
| 3 | 2 | 2017-01-01 | 2 | 2017-02-02 |
| 4 | 2 | 2017-01-01 | 1 | 2017-02-02 |
+-----+-----+-----+-----+
4 rows in set (0.04 sec)

mysql> select B.resolverId as Id from (select creatorId, count(*) as y1 from issue group by creatorId) A right join (select resolverId, count(*) as y2 from issue group by r
esolverId) B on A.creatorId=B.resolverId where ((A.y1<B.y2) or (A.y1 IS NULL and B.y2 is not NULL));
+-----+
| Id |
+-----+
| 2 |
+-----+
1 row in set (0.04 sec)
```