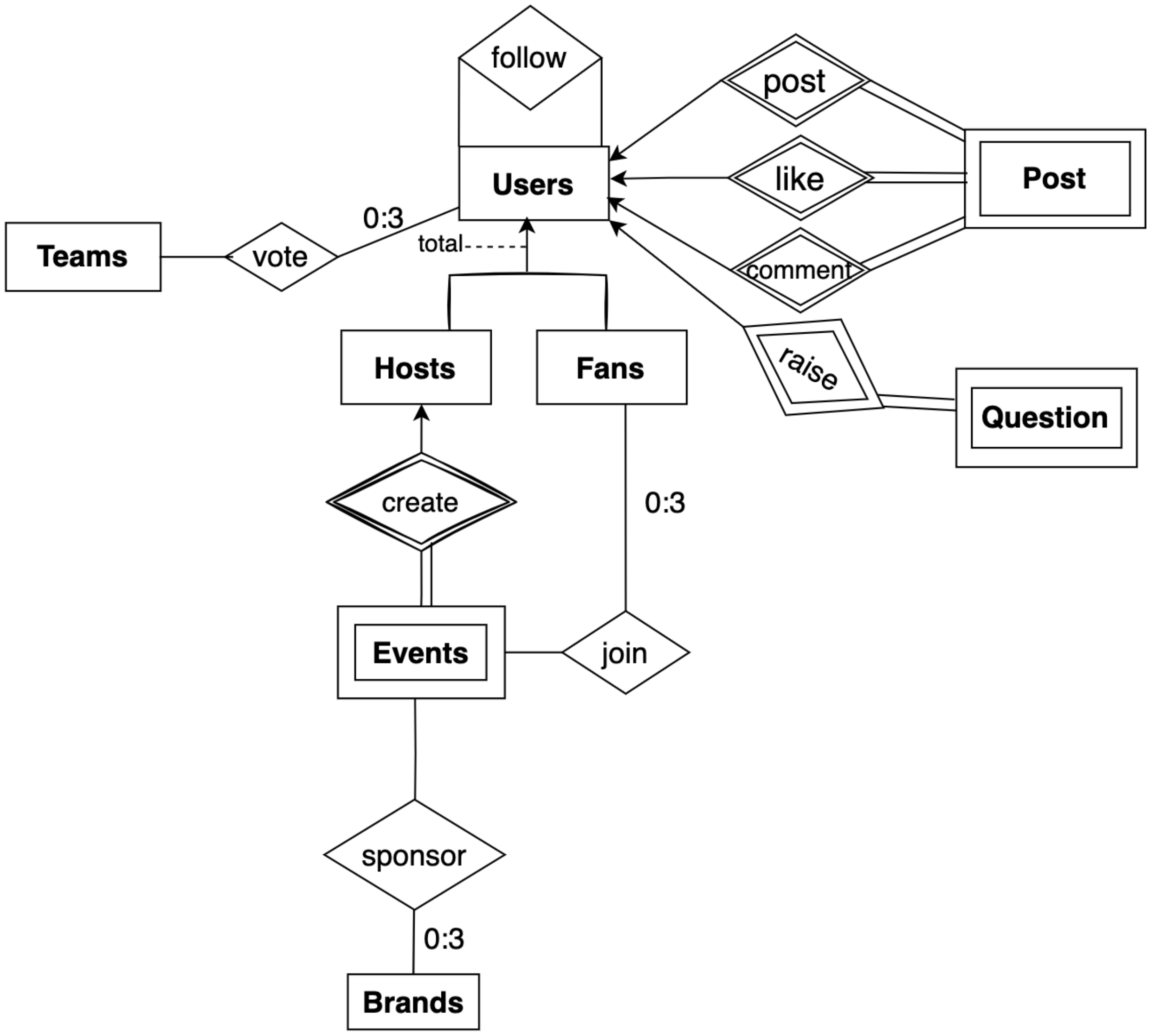
Project 1 Part 2 Report

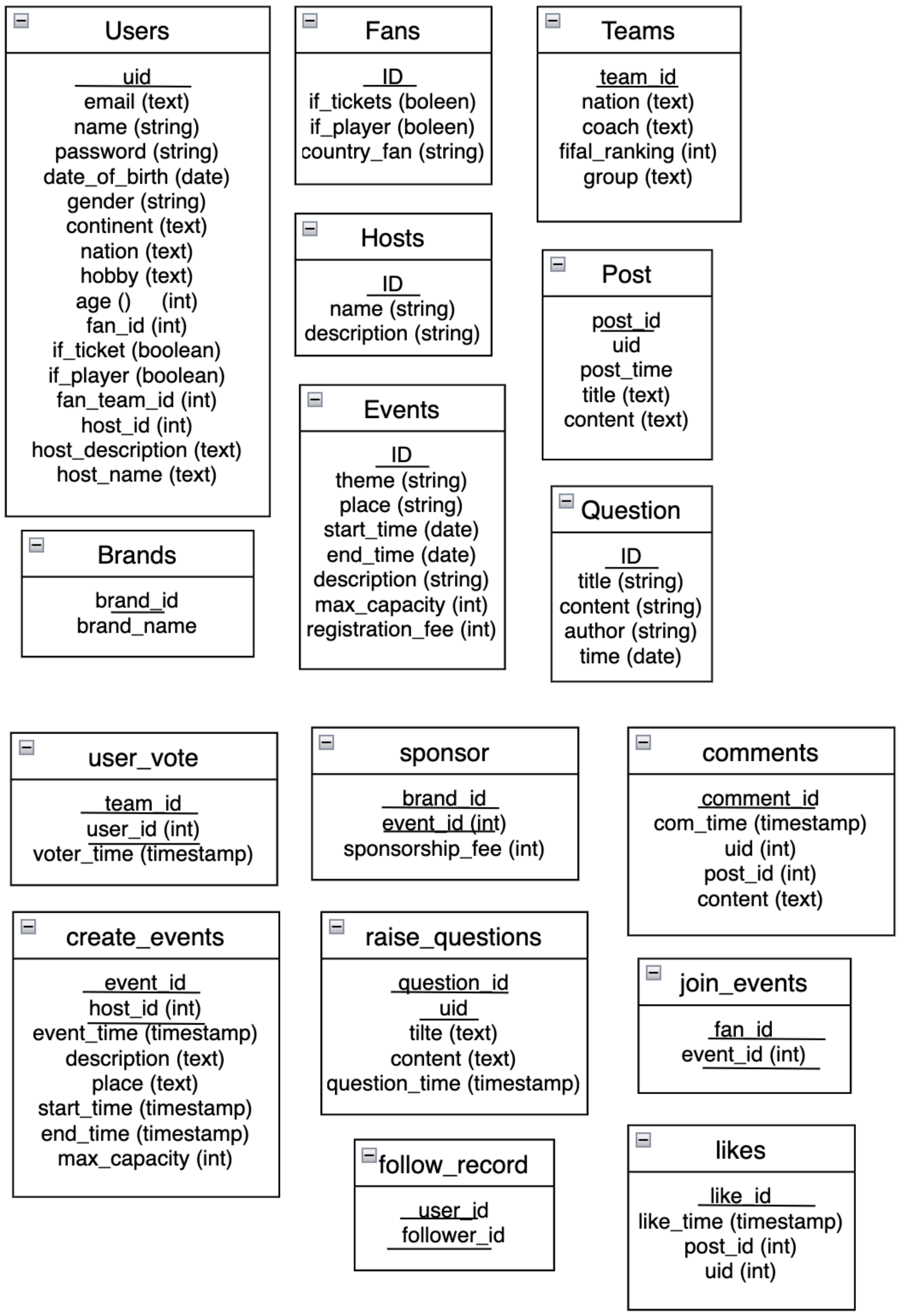
(4771 Introduction to Database)

Shihui Zhu (sz3029), Jie Liu (jl5788)

I. UNI for Schema: sz3029

II. Updated ER Diagram:




Users
uid
email (text)
name (string)
password (string)
date_of_birth (date)
gender (string)
continent (text)
nation (text)
hobby (text)
age () (int)
fan_id (int)
if_ticket (boolean)
if_player (boolean)
fan_team_id (int)
host_id (int)
host_description (text)
host_name (text)
Fans
ID
if_tickets (boleen)
if_player (boleen)
country_fan (string)
Teams
team_id
nation (text)
coach (text)
fifal_ranking (int)
group (text)
Hosts
ID
name (string)
description (string)
Post
post_id
uid
post_time
title (text)
content (text)
Events
ID
theme (string)
place (string)
start_time (date)
end_time (date)
description (string)
max_capacity (int)
registration_fee (int)
Question
ID
title (string)
content (string)
author (string)
time (date)
Brands
brand_id
brand_name
user_vote
team_id
user_id (int)
voter_time (timestamp)
sponsor
brand_id
event_id (int)
sponsorship_fee (int)
comments
comment_id
com_time (timestamp)
uid (int)
post_id (int)
content (text)
create_events
event_id
host_id (int)
event_time (timestamp)
description (text)
place (text)
start_time (timestamp)
end_time (timestamp)
max_capacity (int)
raise_questions
question_id
uid
tilte (text)
content (text)
question_time (timestamp)
join_events
fan_id
event_id (int)
follow_record
user_id
follower_id
likes
like_id
like_time (timestamp)
post_id (int)
uid (int)

III. Updated Schemas

We've made some changes to the entities and relationships, here's the updated ones:

Users:

1. Users are either "**hosts**" or "**fans**" (total participation, this makes more sense), after signing up, they will have globally unique ID and can set their own password Every user should have a unique email address (this enables websites to send their alerts or updates in the future). They can also optionally fill up their personal information, including birth date, gender, nation, ect. al
2. Fans can *post* **posts** to the forum, *like* or *comment* on theirs or others posts to interact on this platform
3. Users can *follow* each other
4. Hosts can *create* **events** that the soccer fans can *join* at most 3 events
5. A user can vote at most 3 times, and can only vote one time for the same team (avoid repetitive voting)

Questions/Q&A:

1. Fans can *send* **questions** like feedback and advice to the webmaster, e.g. raise technical questions etc. al. Each question only has one author.
2. Fans may help *answer* each other's questions in the Q&A section

Events:

1. Events have unique ID
2. Events can have names, descriptions, start and end dates, and max capacity of which the default is 1000 (by setting the names, start and end dates as optional, the dataset can be easier to manage)
3. Event can optionally show its sponsorship information (deleted, the sponsorship information is stored in a different sponsor table)
4. Fans can *join* at most three events

Teams

1. Total 32 national soccer teams which will participate in 2022 Qatar World Cup
2. Teams have unique id, fifa ranking and they can optionally show their coach information

Posts:

1. Posts are *written* or edited by users. A post can only have one author, only the author can edit the post.

2. Users may *like* and *comment* on other's posts.

Brand:

1. Brands have their unique ID, and their names. A brand can *sponsor* at most events (in reality, this constraint encourages more diversity on sponsorships)

Join_event:

1. Remove redundant registration_fee attribute which has been stored in Events table

IV. CREATE commands for schemas:

Google Colab link:
https://colab.research.google.com/drive/1CED_Bi8CoL7UlZ4pMsezUzfYrj3csJuE#scrollTo=C7phEfcKiN1C

1. Entity *teams*
   `teams(team_id, nation, coach, fifa_ranking, group)`

```
CREATE TABLE Teams (
 team_id serial PRIMARY KEY,
 nation TEXT NOT NULL unique,
 coach TEXT not null ,
 fifa_ranking int not null unique check(fifa_ranking>0 and
fifa_ranking < 220 ),
 "group" text not null check ("group" in ('A', 'B', 'C', 'D',
'E', 'F', 'G', 'H'))
);
```

2. Entity *users* (use view for age calculation)
   `users(uid, email, name, password, date_of_birth, gender, continent, nation, hobby, fan_id, if_ticket, if_player, team_id_fan, host_id, host_description, host_name, age)`

```
CREATE TABLE users (
 uid serial PRIMARY KEY, -- user id
 email TEXT UNIQUE NOT NULL,
 name VARCHAR(10) CHECK (name <> ''), -- user name, <= 10 chars
 password VARCHAR(25) NOT NULL, -- user's self-defined password,
<= 25 chars
 date_of_birth DATE, -- user's date of birth
 gender text CHECK (gender IN ('F', 'M', 'Other')),
 continent text CHECK (continent IN ('North America', 'South
America', 'Asia', 'Africa', 'Europe', 'Antartica',
'Australia')),
 nation TEXT,
 hobby TEXT,
  fan_id int unique default null,
 if_ticket BOOLEAN default null,
 if_player BOOLEAN default null,
 fan_team_id int references teams(team_id) default null,

 host_id int unique default null,
 host_description TEXT default null,
 host_name text default null,

 check (((fan_id, if_ticket, if_player, fan_team_id) is null and
(host_id, host_description, host_name) is not null) or
         ((fan_id, if_ticket, if_player, fan_team_id) is not
null and (host_id, host_description, host_name) is null))
);

CREATE OR REPLACE VIEW v_users AS (
 select u.*, date_part('year', age(CURRENT_DATE,
date_of_birth))::int age
 FROM users AS u
);
```

3. Relationship vote

vote relationship between *users* and *teams*: each user can vote for 3 or less different teams (total team voting result obtained from a view function)

`user_vote(team_id, user_id, vote_time)`

```
CREATE TABLE user_vote (
 team_id int REFERENCES teams ON DELETE CASCADE,
 user_id int REFERENCES users(uid) ON DELETE CASCADE,
 vote_time TIMESTAMP DEFAULT current_timestamp,
 PRIMARY KEY (team_id, user_id)
);

CREATE OR REPLACE VIEW v_team_voting_result AS (
 SELECT v.team_id, COUNT(DISTINCT v.user_id) votes_team
 FROM user_vote AS v
 WHERE v.team_id IN (SELECT t.team_id
                     FROM teams t)
 GROUP BY v.team_id
)
CREATE FUNCTION vote_limit() RETURNS trigger AS $vote_limit$
   BEGIN
       -- Check that each fans has <= 3 votes
       IF (SELECT COUNT(f.team_id)
           FROM user_vote f
           WHERE NEW.user_id = f.user_id) > 2 THEN
           RAISE EXCEPTION 'each user can only vote for <= 3
teams';
       END IF;
       RETURN NEW;
   END;
$vote_limit$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS vote_limit ON user_vote CASCADE;
CREATE TRIGGER vote_limit BEFORE INSERT OR UPDATE ON user_vote
```

```
   FOR EACH ROW EXECUTE PROCEDURE vote_limit();
```

4. Relationship create
   create relationship between *hosts* and *events* (a weak entity depending on
   users as hosts). Events have optional registration fee.

   `create_events(event_id, host_id, event_name, description, place, regist_fee, start_time, end_time, max_capacity)`

```
CREATE TABLE create_events (
 event_id serial UNIQUE, -- host can host multiple events
 -- user_id int REFERENCES users(uid) ON DELETE CASCADE,
 host_id int REFERENCES users(host_id) not null,
 event_name TEXT default 'Soccer Party',
 description TEXT,
 place TEXT default 'Qatar',
 regist_fee NUMERIC DEFAULT 0,
 start_time DATE default '2022-11-21',
 end_time DATE default '2022-12-18',
 max_capacity INT DEFAULT 1000,
 PRIMARY KEY (event_id, host_id)
);
```

5. Relationship join (use view for registration fee)
   join relationship between *fans* and *events* - each fan can join at most 3 events

   `join_events(event_id, fan_id, regist_fee)`

```
CREATE TABLE join_events (
 fan_id int REFERENCES users(fan_id),
 event_id int REFERENCES create_events(event_id),
 PRIMARY KEY (event_id,fan_id)
);
```

```
CREATE OR REPLACE VIEW v_join_events AS (
 SELECT j.*, c.regist_fee
 FROM join_events j, create_events c
 WHERE j.event_id = c.event_id
);

CREATE FUNCTION join_limit() RETURNS trigger AS $join_limit$
   BEGIN
       -- Check that each fans has <= 3 votes
       IF (SELECT COUNT(j.event_id)
           FROM join_events j
           WHERE NEW.fan_id = j.fan_id) > 2 THEN
           RAISE EXCEPTION 'fans can only join <= 3 events';
       END IF;
       RETURN NEW;
   END;
$join_limit$ LANGUAGE plpgsql;

CREATE TRIGGER join_limit BEFORE INSERT OR UPDATE ON join_events
   FOR EACH ROW EXECUTE PROCEDURE join_limit();
```

6. Entity *brands*
   brands(<u>brand_id</u>, brand_name, brand_type)

```
CREATE TABLE brands (
 brand_id serial UNIQUE,
 brand_name VARCHAR(50) UNIQUE,
 PRIMARY KEY (brand_id),
 brand_type TEXT DEFAULT 'Others' CHECK (brand_type IN
('Sports', 'Tech', 'Luxury', 'Others'))
)
```

7. Relationship Sponsor

Sponsor relationship between *brands* and *events* - each brands can sponsor 3 or less events

`sponsor(brand_id, event_id, sponsorship_fee)`

```
CREATE TABLE sponsor (
 brand_id INT REFERENCES brands(brand_id),
 event_id INT REFERENCES create_events(event_id),
 sponsorship_fee NUMERIC DEFAULT 0,
 PRIMARY KEY (brand_id, event_id)
)
CREATE FUNCTION sponsor_limit() RETURNS trigger AS
$sponsor_limit$
   BEGIN
       -- Check that each company has <= 3 sponsorships
       IF (SELECT COUNT(s.event_id)
           FROM sponsor s
           WHERE NEW.brand_id = s.brand_id) > 2 THEN
           RAISE EXCEPTION 'companies can only sponsor <= 3
events';
       END IF;
       RETURN NEW;
   END;
$sponsor_limit$ LANGUAGE plpgsql;

CREATE TRIGGER sponsor_limit BEFORE INSERT OR UPDATE ON sponsor
   FOR EACH ROW EXECUTE PROCEDURE sponsor_limit();
```

8. rasie_questions relationship

`raise_questions(question_id, uid, title, content, question_time)`

```
CREATE TABLE raise_questions (
 question_id serial primary key,
 uid int,
 title text not null,
 content text not null,
 question_time timestamp DEFAULT current_timestamp,
 foreign key (uid) references users(uid));
```

9. Entity *posts*

   `posts(post_id, uid, post_time, title, content)`

```
set timezone='America/new_york';
CREATE TABLE Posts(
 post_id serial,
 uid int not null,
 post_time timestamp DEFAULT current_timestamp,
 title text not null,
 content text not null,
 primary key (post_id),
 foreign key (uid) references Users(uid) on delete cascade);
```

10. Likes relationship

    `likes(like_id, like_time, uid)`

```
CREATE TABLE Likes(
 like_id serial primary key,
 like_time timestamp DEFAULT current_timestamp,
 uid int not null,
 foreign key (uid) references Users(uid) on delete cascade);
```

11. Comments relationship

    `comments(comment_id, com_time, uid, content)`

```
CREATE TABLE Comments(
 comment_id serial primary key,
 com_time timestamp DEFAULT current_timestamp,
 uid int not null,
 content text not null,
 foreign key (uid) references Users(uid) on delete cascade);
```

12. Follow relationship

    follow_record(user_id, follower_id)

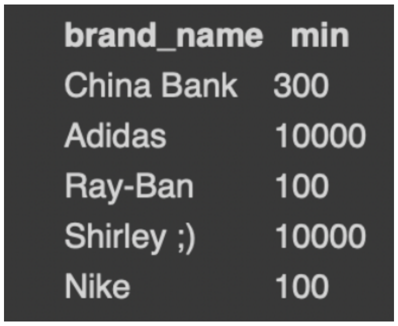
```
create table follow_record(
   user_id int not null references users(uid),
   follower_id int not null references users(uid),
   primary key (user_id, follower_id)
);
```

VI. Interesting queries

1. Description: Select name of brands which have sponsored events and their minimal sponsorship fee

```
SELECT b.brand_name, MIN(s.sponsorship_fee)
FROM sponsor s, brands b
where s.brand_id = b.brand_id
GROUP BY b.brand_name
```

Output:



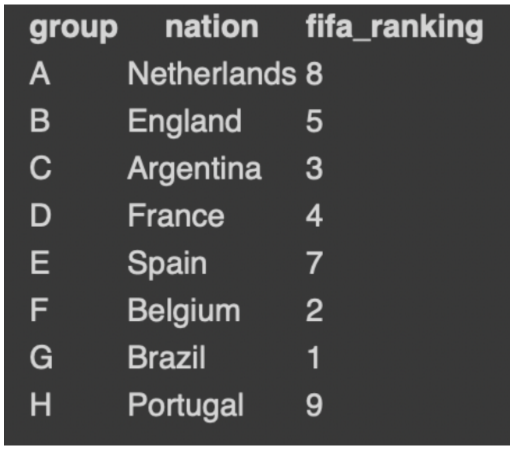
brand_name	min
China Bank	300
Adidas	10000
Ray-Ban	100
Shirley ;)	10000
Nike	100

2. Description: Select the nation, group and FIFA ranking of national soccer teams with the highest ranking among each of eight groups

```
select t."group", t.nation, t.fifa_ranking
```

```
from (select "group",  min(fifa_ranking) ranking
from teams
group by "group") a , teams t
where a.ranking=t.fifa_ranking
order by "group"
```

Output:

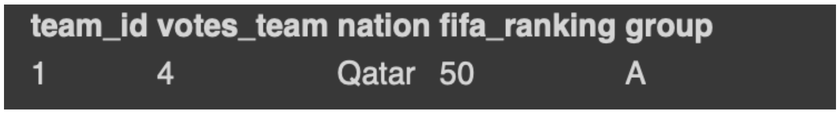


```
group   nation      fifa_ranking
A       Netherlands 8
B       England     5
C       Argentina   3
D       France      4
E       Spain       7
F       Belgium     2
G       Brazil      1
H       Portugal    9
```

3. Description: Select id, votes , nation, FIFA ranking of the most popular (the most votes) team and group of which it belongs to.

```
SELECT v.team_id, votes_team, nation, fifa_ranking, "group"
FROM v_team_voting_result v RIGHT OUTER JOIN teams t
ON v.team_id = t.team_id
WHERE v.votes_team is not null
ORDER BY v.votes_team DESC
limit 1;
```

Output:

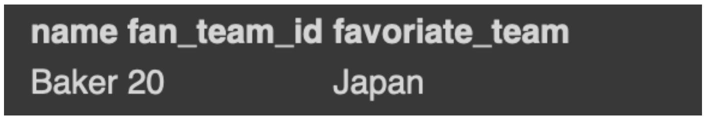


```
team_id votes_team nation fifa_ranking group
1       4          Qatar  50           A
```

4. Description: Select name and favorite national team of soccer fans who vote for their home(favorite) teams

```
select temp1.*, t.nation as favoriate_team
from (SELECT u1.name, u1.fan_team_id
from users u1, user_vote u2
where u1.fan_id is not null and u1.uid = u2.user_id and
u1.fan_team_id = u2.team_id) temp1, teams t
where temp1.fan_team_id = t.team_id
```

Output:



```
name fan_team_id favoriate_team
Baker 20          Japan
```