Introduction to Databases – Project1 Proposal

Bin Qi (bq2130) Yunqiu Yao (yy2827)

1. UNI used on the course DB server: **yy2827**

2. 3 queries:

(1) How many transactions per person in the past week?

```
%%sql
with trans (username, name, trans_id) as (
   select tt.actor_name, vu.full_name, tt.payment_id from venmo_users as vu
   join transa_transf as tt on vu.user_name = tt.actor_name
   where tt.created_time >= current_date - interval '7 days'
   union
   select tt.target_name, vu.full_name, tt.payment_id from venmo_users as vu
   join transa_transf as tt on vu.user_name = tt.target_name
   where tt.created_time >= current_date - interval '7 days'
)
select username, name, count(*) as transactions from trans
group by username, name
order by transactions DESC;
```

- Result table:

	username	name	transactions
0	NoahGershwin	Noah Gershwin	6
1	jjscarz	Julia Scarangella	6
2	christianromeroUP	Christian Romero	5
3	Guo-Zhiqi	Guo Zhiqi	4
4	Marisa-Chambers	Marisa Chambers	2
5	Michael-Caguioa	Michael Caguioa	2
6	elisadangelo	Elisa Dangelo	2
7	Brian-Du-3	Brian Du	2
8	Jeffrey-Kemper-1	Jeffrey Kemper	2
9	Andrew-Kwon-8	Andrew Kwon	1
10	MeganResnick	Megan Resnick	1
11	anthonynguyen1006	Anthony Nguyen	1

- Description: We are interested in computing the amount of transactions one Venmo user participated in (as an actor or target) during the period of data collection. This statistic can be used as one of the risk dimensions. Theoretically, the more the publicized

transactions we could retrieve from one Venmo account, the higher risk it may be exposed to.

(2) How many distinct people one user transacted with in the past week?

```
%%sql
with trans_person (username1, username2) as (
   select tt.actor_name, tt.target_name from transa_transf as tt
   where tt.created_time >= current_date - interval '7 days'
   union
   select tt.target_name, tt.actor_name from transa_transf as tt
   where tt.created_time >= current_date - interval '7 days'
)
select username1, vu.full_name as name, count(username1) as persons from trans_person as tp
join venmo_users as vu on vu.user_name = tp.username1
group by username1, name
order by persons DESC;
```

- Result table:

	username1	name	persons
0	NoahGershwin	Noah Gershwin	5
1	jjscarz	Julia Scarangella	5
2	christianromeroUP	Christian Romero	4
3	Guo-Zhiqi	Guo Zhiqi	4
4	elisadangelo	Elisa Dangelo	2
5	Brian-Du-3	Brian Du	2
6	Marisa-Chambers	Marisa Chambers	1
7	Andrew-Kwon-8	Andrew Kwon	1
8	anthonynguyen1006	Anthony Nguyen	1
9	Michael-Caguioa	Michael Caguioa	1
10	Jeffrey-Kemper-1	Jeffrey Kemper	1
11	MeganResnick	Megan Resnick	1

- Description: This calculates the size of social network shown in transactions for each Venmo user. The larger the value, the higher the risk of privacy leakage.

(3) The largest length of message for each Venmo user's transactions

```
%%sql
drop table if exists mes cascade;
create table mes (username, name, trans_id, message) as (
```

```
    select tt.actor_name, vu.full_name, tt.payment_id, tt.message from venmo_users as
vu
    join transa_transf as tt on vu.user_name = tt.actor_name
    where tt.created_time >= current_date - interval '7 days'
);
drop function if exists strlen cascade;
CREATE FUNCTION strlen(mes) RETURNS integer AS $$
    SELECT char_length($1.message);
$$ LANGUAGE SQL;
select username, name, max(strlen(mes.*)) as length from mes
group by username, name
order by length DESC;
```

- Result table:

	username	name	length
0	Brian-Du-3	Brian Du	99
1	NoahGershwin	Noah Gershwin	31
2	jjscarz	Julia Scarangella	19
3	MeganResnick	Megan Resnick	18
4	christianromeroUP	Christian Romero	14
5	Guo-Zhiqi	Guo Zhiqi	13
6	Jeffrey-Kemper-1	Jeffrey Kemper	10
7	Marisa-Chambers	Marisa Chambers	10
8	Michael-Caguioa	Michael Caguioa	7
9	elisadangelo	Elisa Dangelo	1

- Description: We use the length of “messages” under transaction as another risk factor. This captures the detailedness of notes one user would like to publicize when transacting on Venmo. More characters indicate more info leakage.

3. Changes made since last submission

   Based on the mentor’s feedback, we revised the “interact” relationship part of E-R diagram, simplifying it into 2 separate relationships: Comments and Like (thanks to TA Ivy Chen).

   Also, we added a summarized cache table “Risk_Rank” (thanks to TA Ivy Chen and our mentors’ advice). We revised the corresponding

“relationships&constraints” part. More details can be found in the revised proposal below.

To fulfill the requirement of entity/relationship numbers, we added a new entity “Bank_acciunts”, to record credit card info associated with each Venmo account. It is also a weak entity for “Venmo_users”.

We added 2 “check” constraints in the tables “transa_transf” and “bank_pay”.

The summarized cache table “Risk_Rank”, which will provide our app user a risk score and some google search results, has not yet been completed. We plan to discuss with our mentor in the first place, then enrich more queries to generate this application-level table.

(We attach the revised project proposal below, and look forward to your feedbacks and corrections. Thank you!)

General:

Our application tries to monitor the private information leakage on Venmo, a popular peer-to-peer payment app with social media features. There are tons of transactions between different users in our app as Venmo displays private data in a way more public than we could imagine. With our app, a user could track her transaction history and the extent of personal data publicized by Venmo. Also, she could search for someone else's personal/social/financial info on Venmo that she is interested in. While free and convenient to use, Venmo users could be extremely vulnerable to info leakage. Our app can be a timely alert that some Venmo users should change their security settings and using behavior in order not to fall for potential online scams.

To each user, by inputting a specific Venmo username, our app could generate a risk level indicating the comprehensive safety status of that Venmo account. The domain of "risk level" is {0,1,2,3,4}, and the higher the value, the riskier. We calculate this score by aggregating various dimensions of privacy i.e., the volume of public transaction history; the size of social network shown in transactions and interactions; any accessible connection to his/her Facebook account; the length of message attached to each transaction. More precisely:

rid	risk_level	public_transaction>5	friends>3	link_to_Facebook	Notes_length>10	Google_search_results
1	0	0	0	0	0	null
2	1	1	0	0	0	venmo+public+risk
3	1	0	1	0	0	venmo+friends+risk
4	1	0	0	1	0	venmo+facebook+risk
5	1	0	0	0	1	venmo+message+risk
6	2	1	1	0	0	public+venmo+friends+risk
7	2	1	0	1	0	public+venmo+facebook+risk
8	2	1	0	0	1	public+venmo+message+risk
9	2	0	1	1	0	venmo+friends+facebook+risk
10	2	0	1	0	1	venmo+friends+message+risk
11	2	0	0	1	1	venmo+facebook+message+risk
12	3	1	1	1	0	public+venmo+friends+facebook+risk
13	3	0	1	1	1	venmo+friends+facebook+message+risk
14	3	1	0	1	1	public+venmo+facebook+message+risk
15	3	1	1	0	1	public+venmo+friends+message+risk
16	4	1	1	1	1	public+venmo+friends+facebook+message+risk

Furthermore, relevant google search results regarding the potential risk factors can also be posted in our app to inform and remind our users. The

last column of the table above lists the key words we may apply in a UDF query which presents the risk related google search results.

It may appear that the table “Risk_Rank” is sort of redundant, as each attribute‘s value can be generated by aggregating other tables. However, from the perspective of app use, as this application-level table may not be updated as frequently other (risk levels will be updated once a week, while other data can be daily updated) it seems necessary to include a cache table containing “delayed” risk levels in our database so that an existent risk score can be accessed any time by our user.

Data source:

We will obtain the transaction feed and personal data via Venmo public API. It appears that Venmo is updating the transactions marked as “public” dynamically to the website. Scraping those publicly available data from the site, we can see all the information about the transactions and the initiators and respondents, such as their full names, photos, transaction time, message, etc. In other words, all the private information that we value is readily shared with the rest of the world.

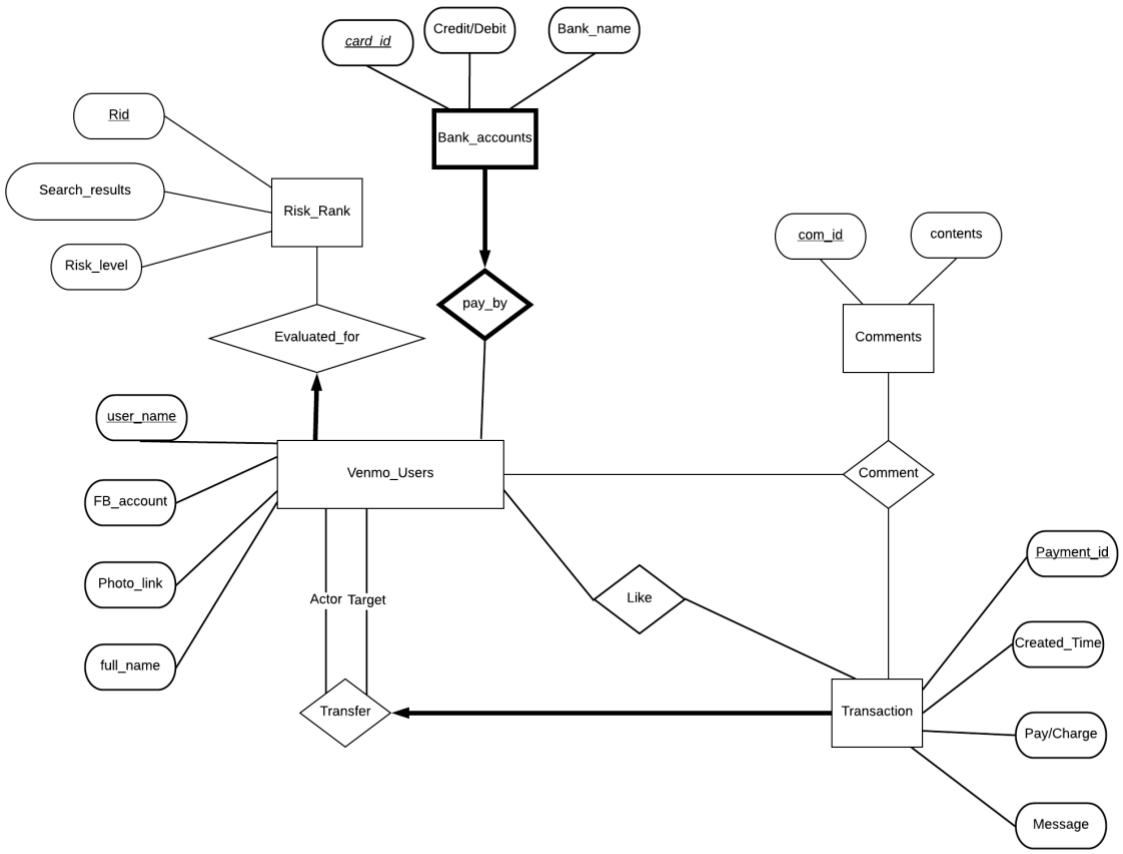
Examples:

Entities: Venmo_Users (user_name, full_name, FB_link, Photo_link), Transaction (Payment_id, Pay/Charge, Message, Create_Time), Comments (com_id, contents)
Risk_Rank (rid, Risk_level, Search_results)
Bank_accounts (card_id, Credit/Debit, Bank_name)

Relationship & constraints:
Each transaction has exactly one actor (user) and one target (user); a Venmo user can comment on several transactions and one transaction for several times; a Venmo user can “like” a transaction only once; A Venmo account can relate to several bank accounts, while each bank account is

connected to exactly one Venmo user and is its weak entity; Each user has exactly one assigned risk rank.

E-R diagram:



Schema:

```
create table risk_rank(
    rid int primary key,
    risk_level int not null,
    search_result text not null
);
```

```
create table venmo_users(
    user_name text primary key,
    full_name text not null,
    photo_link text,
    fb_account text
);

create table transa_transf(
    payment_id int primary key,
    created_time date not null,
    pay_charge text not null check (pay_charge = 'pay' or pay_charge ='charge'),
    actor_name text not null references venmo_users(user_name),
    target_name text not null references venmo_users(user_name),
    message text not null
);

create table likes(
    payment_id int references transa_transf,
    user_name text references venmo_users,
    primary key(payment_id,user_name)
);

create table comments(
    com_id int primary key,
    contents text not null
);

create table comment(
    com_id int references comments,
    payment_id int references transa_transf,
    user_name text references venmo_users,
    primary key(com_id, payment_id,user_name)

);

create table bank_pay(
    card_id int primary key,
    user_name text not null references venmo_users on delete cascade,
    credit_debit text not null check (credit_debit = 'credit' or credit_debit ='debit'),
    bank_name text not null
);

create table venmo_users_evaluate(
```

```
    user_name text primary key,
    full_name text not null,
    photo_link text,
    fb_account text,
    rid int not null references risk_rank
);
```

In [1]:

```
ib.connect_db("ib://Bin_Qi/W4111/fs/Instabase Drive/databases")
```

```
Connected to: ib://Bin_Qi/W4111/fs/Instabase Drive/databases
```

In [2]:

```
%config SqlMagic.autocommit=False
ib.connect_db("postgresql://student:w4111student@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/w4111")
```

```
Connected to: postgresql://student:w4111student@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/w4111
```

risk_rank table

In [3]:

```
%%sql

drop table if exists risk_rank cascade;

create table risk_rank(
    rid int primary key,
    risk_level int not null,
    search_result text not null
);
```

```
Done.
Done.
```

Out[3]:

□

venmo users

In [4]:

```
%%sql

drop table if exists venmo_users cascade;

create table venmo_users(
    user_name text primary key,
    full_name text not null,
    photo_link text,
    fb_account text
);
```

```
Done.
Done.
```

Out[4]:

□

transaction_transfer

In [5]:

```
%%sql

drop table if exists transa_transf cascade;

create table transa_transf(
    payment_id int primary key,
    created_time date not null,
    pay_charge text not null check (pay_charge = 'pay' or pay_charge ='charge'),
    actor_name text not null references venmo_users(user_name),
    target_name text not null references venmo_users(user_name),
    message text not null
);
```

```
Done.
Done.
```

Out[5]:

□

likes

In [6]:

```
%%sql
drop table if exists likes cascade;

create table likes(
    payment_id int references transa_transf,
    user_name text references venmo_users,
    primary key(payment_id,user_name)
);
```

Done.
Done.

Out[6]:

□

comments

In [7]:

```
%%sql
drop table if exists comment, comments cascade;

create table comments(
    com_id int primary key,
    contents text not null
);

create table comment(
    com_id int references comments,
    payment_id int references transa_transf,
    user_name text references venmo_users,
    primary key(com_id, payment_id,user_name)
);
```

Done.
Done.
Done.

Out[7]:

□

Bank_pay (weak entity for Venmo users)

In [8]:

```
%%sql
drop table if exists bank_pay;

create table bank_pay(
    card_id int primary key,
    user_name text not null references venmo_users on delete cascade,
    credit_debit text not null check (credit_debit = 'credit' or credit_debit ='
debit'),
    bank_name text not null
);
```

```
Done.
Done.
```

Out[8]:

□

Populate the tables

risk_rank

In [9]:

```
%%sql
insert into risk_rank VALUES
(1, 0, ''),
(2, 1, 'venmo+public+risk'),
(3, 1, 'venmo+friends+risk'),
(4, 1, 'venmo+facebook+risk'),
(5, 1, 'venmo+message+risk'),
(6, 2, 'public+venmo+friends+risk'),
(7, 2, 'public+venmo+facebook+risk'),
(8, 2, 'public+venmo+message+risk'),
(9, 2, 'venmo+friends+facebook+risk'),
(10, 2, 'venmo+friends+message+risk'),
(11, 2, 'venmo+facebook+message+risk'),
(12, 3, 'public+venmo+friends+facebook+risk'),
(13, 3, 'venmo+friends+facebook+message+risk'),
(14, 3, 'public+venmo+facebook+message+risk'),
(15, 3, 'public+venmo+friends+message+risk'),
(16, 4, 'public+venmo+friends+facebook+message+risk');
```

```
16 rows affected.
```

Out[9]:

□

In [10]:

```
%%sql
select* from risk_rank
```

16 rows affected.

Out[10]:

	rid	risk_level	search_result
0	1	0	
1	2	1	venmo+public+risk
2	3	1	venmo+friends+risk
3	4	1	venmo+facebook+risk
4	5	1	venmo+message+risk
5	6	2	public+venmo+friends+risk
6	7	2	public+venmo+facebook+risk
7	8	2	public+venmo+message+risk
8	9	2	venmo+friends+facebook+risk
9	10	2	venmo+friends+message+risk
10	11	2	venmo+facebook+message+risk
11	12	3	public+venmo+friends+facebook+risk
12	13	3	venmo+friends+facebook+message+risk
13	14	3	public+venmo+facebook+message+risk
14	15	3	public+venmo+friends+message+risk
15	16	4	public+venmo+friends+facebook+message+risk

venmo_users

In [11]:

```
%%sql
insert into venmo_users VALUES
( 'christianromeroUP', 'Christian Romero', 'https://graph.facebook.com/v2.10/102
05730053050604/picture?type=large', '10205730053050604'),
( 'Jeffrey-Kemper-1', 'Jeffrey Kemper',  'https://venmopics.appspot.com/u/v1/m/c
50b130d-feca-4dac-8caa-0961294bdd36', ''),
(  'Guo-Zhiqi', 'Guo Zhiqi', 'https://venmopics.appspot.com/u/v1/n/0074e266-68ff
-461b-96b5-9bb19661ec06', ''),
(  'MeganResnick', 'Megan Resnick',  'https://venmopics.appspot.com/u/v3/n/e9692
dd2-5a9f-4b1d-99b8-b074144b82ee', ''),
(  'NoahGershwin', 'Noah Gershwin', 'https://venmopics.appspot.com/u/v1/m/09298c
d9-cfb7-46b2-a364-cf59d505c0a4', ''),
( 'Marisa-Chambers', 'Marisa Chambers', 'https://graph.facebook.com/v2.10/102073
15721424646/picture?type=large', '10207315721424646'),
(  'elisadangelo', 'Elisa Dangelo', 'https://graph.facebook.com/v2.10/7425877525
39110/picture?type=large', '742587752539110'),
( 'anthonynguyen1006', 'Anthony Nguyen', 'https://venmopics.appspot.com/u/v2/m/f
4d64bee-fcf6-4d50-932d-5f0d77875727', ''),
(  'jjscarz', 'Julia Scarangella', 'https://venmopics.appspot.com/u/v1/m/d9ccd08
6-8cf3-4a1e-99ae-9bd12c1e67ee', ''),
(  'Michael-Caguioa', 'Michael Caguioa',  'https://venmopics.appspot.com/u/v2/m/
06b44f04-063a-4dc5-aa9d-410397fd023f', ''),
( 'Brian-Du-3', 'Brian Du', 'https://venmopics.appspot.com/u/v1/n/cba5b23a-940d-
4d2f-8df0-631d9a96f09e', ''),
( 'Andrew-Kwon-8', 'Andrew Kwon', 'https://venmopics.appspot.com/u/v1/n/c66f063d
-5e59-4ba3-a766-0e673323326d', '');
```

12 rows affected.

Out[11]:

In [12]:

```
%%sql
select* from venmo_users
```

```
12 rows affected.
```

Out[12]:

	user_name	full_name	photo_link
0	christianromeroUP	Christian Romero	https://graph.facebook.com/v2.10/1020573005305...
1	Jeffrey-Kemper-1	Jeffrey Kemper	https://venmopics.appspot.com/u/v1/m/c50b130d-...
2	Guo-Zhiqi	Guo Zhiqi	https://venmopics.appspot.com/u/v1/n/0074e266-...
3	MeganResnick	Megan Resnick	https://venmopics.appspot.com/u/v3/n/e9692dd2-...
4	NoahGershwin	Noah Gershwin	https://venmopics.appspot.com/u/v1/m/09298cd9-...
5	Marisa-Chambers	Marisa Chambers	https://graph.facebook.com/v2.10/1020731572142...
6	elisadangelo	Elisa Dangelo	https://graph.facebook.com/v2.10/7425877525391...
7	anthonynguyen1006	Anthony Nguyen	https://venmopics.appspot.com/u/v2/m/f4d64bee-...
8	jjscarz	Julia Scarangella	https://venmopics.appspot.com/u/v1/m/d9ccd086-...
9	Michael-Caguioa	Michael Caguioa	https://venmopics.appspot.com/u/v2/m/06b44f04-...
10	Brian-Du-3	Brian Du	https://venmopics.appspot.com/u/v1/n/cba5b23a-...
11	Andrew-Kwon-8	Andrew Kwon	https://venmopics.appspot.com/u/v1/n/c66f063d-...

transa_transf

In [13]:

```
%%sql
insert into transa_transf VALUES
(1239057877, '2018-10-22', 'pay', 'christianromeroUP', 'Jeffrey-Kemper-1', 'desk
'),
(1239057876, '2018-10-22',  'pay', 'Jeffrey-Kemper-1', 'christianromeroUP',  'Th
anks bud'),
(1239057871, '2018-10-22', 'charge', 'Guo-Zhiqi', 'christianromeroUP', 'Frothy m
onkey'),
(1239057870, '2018-10-21',  'pay', 'MeganResnick', 'christianromeroUP', 'Chupand
o mis tetas'),
(1239057872, '2018-10-21', 'charge', 'christianromeroUP', 'NoahGershwin', 'Love
u ❤️☺️😩👌🏻'),
(1239057865, '2018-10-21', 'charge', 'NoahGershwin', 'Marisa-Chambers', '🍕'),
(1239057863, '2018-10-21', 'pay', 'Marisa-Chambers', 'NoahGershwin', 'Chipotle
🌯'),
(1239057859, '2018-10-20', 'charge', 'elisadangelo', 'NoahGershwin', '🍼'),
(1239057846, '2018-10-20', 'charge', 'NoahGershwin', 'anthonynguyen1006', 'Uber
from Nash airport to hotel'),
(1239057848, '2018-10-20',  'pay', 'jjscarz', 'NoahGershwin', 'Security deposit
💵⬅️BACK'),
(1239057849, '2018-10-20', 'charge', 'Michael-Caguioa', 'jjscarz','🍬🤱'),
(1239057850, '2018-10-20', 'pay', 'jjscarz', 'Brian-Du-3', 'Lyft+ decorations🌟'
),
(1239057853, '2018-10-20', 'pay', 'Michael-Caguioa', 'jjscarz', 'Karaoke'),
(1239057836, '2018-10-19', 'pay', 'jjscarz', 'Andrew-Kwon-8', 'U know what ;)'),
(1239057840, '2018-10-19', 'charge', 'Guo-Zhiqi', 'elisadangelo', 'Tito's'),
(1239057842, '2018-10-19', 'charge', 'Brian-Du-3','Guo-Zhiqi', 'Im doing hw at 3
am and I just wanna go to my balcony and listen to sad music thnx for the grocer
ies'),
(1239057832, '2018-10-19', 'pay', 'Guo-Zhiqi', 'jjscarz', 'Ur the best');
```

17 rows affected.

Out[13]:

In [14]:

```
%%sql
select* from transa_transf
```

17 rows affected.

Out[14]:

	payment_id	created_time	pay_charge	actor_name	target_name	m
0	1239057877	2018-10-22	pay	christianromeroUP	Jeffrey-Kemper-1	desk
1	1239057876	2018-10-22	pay	Jeffrey-Kemper-1	christianromeroUP	Thar

2	1239057871	2018-10-22	charge	Guo-Zhiqi	christianromeroUP	Frot mor
3	1239057870	2018-10-21	pay	MeganResnick	christianromeroUP	Chu mis
4	1239057872	2018-10-21	charge	christianromeroUP	NoahGershwin	Love ☺️😘
5	1239057865	2018-10-21	charge	NoahGershwin	Marisa-Chambers	🍕
6	1239057863	2018-10-21	pay	Marisa-Chambers	NoahGershwin	Chip 🌯
7	1239057859	2018-10-20	charge	elisadangelo	NoahGershwin	🍼
8	1239057846	2018-10-20	charge	NoahGershwin	anthonynguyen1006	Ube Nas airp hote
9	1239057848	2018-10-20	pay	jjscarz	NoahGershwin	Sec dep ⬅ BACK
10	1239057849	2018-10-20	charge	Michael-Caguioa	jjscarz	🍬🎉
11	1239057850	2018-10-20	pay	jjscarz	Brian-Du-3	Lyft- dec ⭐
12	1239057853	2018-10-20	pay	Michael-Caguioa	jjscarz	Kara
13	1239057836	2018-10-19	pay	jjscarz	Andrew-Kwon-8	U kr wha
14	1239057840	2018-10-19	charge	Guo-Zhiqi	elisadangelo	Tito'
15	1239057842	2018-10-19	charge	Brian-Du-3	Guo-Zhiqi	Im c hw a and wan to m
16	1239057832	2018-10-19	pay	Guo-Zhiqi	jjscarz	Ur th

Bank_pay (weak entity for Venmo users)

In [15]:

```
%%sql
insert into bank_pay VALUES
(123971, 'christianromeroUP', 'credit', 'Chase'),
(123962, 'christianromeroUP',  'debit', 'BOA'),
(123913, 'Jeffrey-Kemper-1', 'credit', 'Citi'),
(123904, 'Guo-Zhiqi',  'credit', 'Discover'),
(123925, 'MeganResnick', 'debit', 'BNP'),
(123956, 'NoahGershwin', 'credit', 'RBC'),
(123937, 'Marisa-Chambers', 'credit', 'Chase'),
(123998, 'elisadangelo', 'debit', 'Citi'),
(123969, 'anthonynguyen1006', 'credit', 'BOA'),
(123980, 'jjscarz',  'credit', 'Chase'),
(123991, 'jjscarz', 'debit', 'DB'),
(123902, 'Michael-Caguioa', 'debit', 'Chase'),
(123933, 'Brian-Du-3', 'credit', 'HSBC'),
(123964, 'Andrew-Kwon-8', 'credit', 'Discover')
```

```
14 rows affected.
```

Out[15]:

□

In [16]:

```
%%sql
select * from bank_pay;
```

```
14 rows affected.
```

Out[16]:

	card_id	user_name	credit_debit	bank_name
0	123971	christianromeroUP	credit	Chase
1	123962	christianromeroUP	debit	BOA
2	123913	Jeffrey-Kemper-1	credit	Citi
3	123904	Guo-Zhiqi	credit	Discover
4	123925	MeganResnick	debit	BNP
5	123956	NoahGershwin	credit	RBC
6	123937	Marisa-Chambers	credit	Chase
7	123998	elisadangelo	debit	Citi
8	123969	anthonynguyen1006	credit	BOA
9	123980	jjscarz	credit	Chase
10	123991	jjscarz	debit	DB
11	123902	Michael-Caguioa	debit	Chase
12	123933	Brian-Du-3	credit	HSBC
13	123964	Andrew-Kwon-8	credit	Discover

likes

In [17]:

```
%%sql
insert into likes(payment_id,user_name) values
(1239057877,'christianromeroUP'),
(1239057877,'jjscarz'),
(1239057877,'Brian-Du-3'),
(1239057876,'Brian-Du-3'),
(1239057876,'Guo-Zhiqi'),
(1239057876,'NoahGershwin'),
(1239057876,'anthonynguyen1006'),
(1239057876,'christianromeroUP'),
(1239057876,'jjscarz'),
(1239057876,'Andrew-Kwon-8'),
(1239057876,'Marisa-Chambers'),
(1239057871,'Marisa-Chambers'),
(1239057871,'Jeffrey-Kemper-1'),
(1239057870,'Guo-Zhiqi'),
(1239057872,'Jeffrey-Kemper-1'),
(1239057872,'elisadangelo'),
(1239057865,'MeganResnick'),
(1239057865,'Brian-Du-3'),
(1239057865,'Marisa-Chambers'),
(1239057865,'Michael-Caguioa'),
(1239057863,'Jeffrey-Kemper-1'),
(1239057863,'christianromeroUP'),
(1239057863,'MeganResnick'),
(1239057863,'elisadangelo'),
(1239057846,'Jeffrey-Kemper-1'),
(1239057848,'MeganResnick'),
(1239057848,'anthonynguyen1006'),
(1239057850,'Guo-Zhiqi'),
(1239057850,'NoahGershwin'),
(1239057850,'anthonynguyen1006'),
(1239057853,'Michael-Caguioa'),
(1239057853,'jjscarz'),
(1239057853,'elisadangelo'),
(1239057853,'Jeffrey-Kemper-1'),
(1239057836,'Andrew-Kwon-8'),
(1239057836,'MeganResnick'),
(1239057840,'Andrew-Kwon-8'),
(1239057840,'NoahGershwin'),
(1239057842,'elisadangelo'),
(1239057842,'christianromeroUP'),
(1239057832,'Michael-Caguioa')
;
```

```
41 rows affected.
```

Out[17]:

□

In [18]:

```
%%sql
select distinct(payment_id) from likes;
```

15 rows affected.

Out[18]:

	payment_id
0	1239057840
1	1239057872
2	1239057863
3	1239057871
4	1239057850
5	1239057836
6	1239057848
7	1239057870
8	1239057876
9	1239057832
10	1239057865
11	1239057877
12	1239057846
13	1239057842
14	1239057853

comments

In [19]:

```
%%sql
insert into comments values
(1,'hahaha'),
(2,'cute'),
(3,'best'),
(4,'what?'),
(5,'so hardworking'),
(6,'omg take care'),
(7,'decorations for xmas?'),
(8,'sounds awsome'),
(9,'oh which one?'),
(10,'love chipotle'),
(11,'seems delicious'),
(12,'wow'),
(13,'Travel lol'),
(14,'love uuuuuu'),
(15,'no problem')
 ;
```

```
15 rows affected.
```

Out[19]:

□

In [20]:

```
%sql select * from comments
```

```
15 rows affected.
```

```
Out[20]:
```

	com_id	contents
0	1	hahaha
1	2	cute
2	3	best
3	4	what?
4	5	so hardworking
5	6	omg take care
6	7	decorations for xmas?
7	8	sounds awsome
8	9	oh which one?
9	10	love chipotle
10	11	seems delicious
11	12	wow
12	13	Travel lol
13	14	love uuuuuu
14	15	no problem

comment

In [21]:

```
%%sql
insert into comment values
(3,1239057832,'MeganResnick'),
(4,1239057836,'Brian-Du-3'),
(5,1239057842,'Guo-Zhiqi'),
(6,1239057842,'Andrew-Kwon-8'),
(1,1239057870,'christianromeroUP'),
(2,1239057849,'jjscarz'),
(7,1239057850,'Brian-Du-3'),
(8,1239057853,'Michael-Caguioa'),
(9,1239057853,'anthonynguyen1006'),
(2,1239057859,'elisadangelo'),
(10,1239057863,'Jeffrey-Kemper-1'),
(11,1239057865,'Marisa-Chambers'),
(12,1239057846,'jjscarz'),
(13,1239057846,'NoahGershwin'),
(1,1239057846,'jjscarz'),
(14,1239057872,'NoahGershwin'),
(15,1239057876,'christianromeroUP'),
(1,1239057876,'MeganResnick')
;
```

18 rows affected.

Out[21]:

In [22]:

```
%sql select * from comment;
```

18 rows affected.

Out[22]:

	com_id	payment_id	user_name
0	3	1239057832	MeganResnick
1	4	1239057836	Brian-Du-3
2	5	1239057842	Guo-Zhiqi
3	6	1239057842	Andrew-Kwon-8
4	1	1239057870	christianromeroUP
5	2	1239057849	jjscarz
6	7	1239057850	Brian-Du-3
7	8	1239057853	Michael-Caguioa
8	9	1239057853	anthonynguyen1006
9	2	1239057859	elisadangelo
10	10	1239057863	Jeffrey-Kemper-1
11	11	1239057865	Marisa-Chambers
12	12	1239057846	jjscarz
13	13	1239057846	NoahGershwin
14	1	1239057846	jjscarz
15	14	1239057872	NoahGershwin
16	15	1239057876	christianromeroUP
17	1	1239057876	MeganResnick

Queries

1. How many transactions per person in the past week?

In [23]:

```
%%sql
with trans (username, name, trans_id) as (
    select tt.actor_name, vu.full_name, tt.payment_id from venmo_users as vu
    join transa_transf as tt on vu.user_name = tt.actor_name
    where tt.created_time >= current_date - interval '7 days'
    union
    select tt.target_name, vu.full_name, tt.payment_id from venmo_users as vu
    join transa_transf as tt on vu.user_name = tt.target_name
    where tt.created_time >= current_date - interval '7 days'
)
select username, name, count(*) as transactions from trans
group by username, name
order by transactions DESC;
```

12 rows affected.

Out[23]:

	username	name	transactions
0	NoahGershwin	Noah Gershwin	6
1	jjscarz	Julia Scarangella	6
2	christianromeroUP	Christian Romero	5
3	Guo-Zhiqi	Guo Zhiqi	4
4	Marisa-Chambers	Marisa Chambers	2
5	Michael-Caguioa	Michael Caguioa	2
6	elisadangelo	Elisa Dangelo	2
7	Brian-Du-3	Brian Du	2
8	Jeffrey-Kemper-1	Jeffrey Kemper	2
9	Andrew-Kwon-8	Andrew Kwon	1
10	MeganResnick	Megan Resnick	1
11	anthonynguyen1006	Anthony Nguyen	1

2. How many distinct people one user transacted with in the past week?

In [24]:

```
%%sql
with trans_person (username1, username2) as (
    select tt.actor_name, tt.target_name from transa_transf as tt
    where tt.created_time >= current_date - interval '7 days'
    union
    select tt.target_name, tt.actor_name from transa_transf as tt
    where tt.created_time >= current_date - interval '7 days'
)
select username1, vu.full_name as name, count(username1) as persons from trans_p
erson as tp
join venmo_users as vu on vu.user_name = tp.username1
group by username1, name
order by persons DESC;
```

12 rows affected.

Out[24]:

	username1	name	persons
0	NoahGershwin	Noah Gershwin	5
1	jjscarz	Julia Scarangella	5
2	christianromeroUP	Christian Romero	4
3	Guo-Zhiqi	Guo Zhiqi	4
4	elisadangelo	Elisa Dangelo	2
5	Brian-Du-3	Brian Du	2
6	Marisa-Chambers	Marisa Chambers	1
7	Andrew-Kwon-8	Andrew Kwon	1
8	anthonynguyen1006	Anthony Nguyen	1
9	Michael-Caguioa	Michael Caguioa	1
10	Jeffrey-Kemper-1	Jeffrey Kemper	1
11	MeganResnick	Megan Resnick	1

3. The largest length of message for each Venmo user's transactions

In [25]:

```
%%sql
drop table if exists mes cascade;
create table mes (username, name, trans_id, message) as (
    select tt.actor_name, vu.full_name, tt.payment_id, tt.message from venmo_use
rs as vu
    join transa_transf as tt on vu.user_name = tt.actor_name
    where tt.created_time >= current_date - interval '7 days'
);
drop function if exists strlen cascade;
CREATE FUNCTION strlen(mes) RETURNS integer AS $$
    SELECT char_length($1.message);
$$ LANGUAGE SQL;
select username, name, max(strlen(mes.*)) as length from mes
group by username, name
order by length DESC;
```

```
Done.
17 rows affected.
Done.
Done.
10 rows affected.
```

Out[25]:

	username	name	length
0	Brian-Du-3	Brian Du	99
1	NoahGershwin	Noah Gershwin	31
2	jjscarz	Julia Scarangella	19
3	MeganResnick	Megan Resnick	18
4	christianromeroUP	Christian Romero	14
5	Guo-Zhiqi	Guo Zhiqi	13
6	Jeffrey-Kemper-1	Jeffrey Kemper	10
7	Marisa-Chambers	Marisa Chambers	10
8	Michael-Caguioa	Michael Caguioa	7
9	elisadangelo	Elisa Dangelo	1

The final Application-level result table (will be finished later)

venmo users with risk levels

In [26]:

```
%%sql

drop table if exists venmo_users_evaluate cascade;

create table venmo_users_evaluate(
    user_name text primary key,
    full_name text not null,
    photo_link text,
    fb_account text,
    rid int not null references risk_rank
);
```

Done.
Done.

Out[26]: