a. The SQL statements used to define and create tables forming your relational database schema. Include all FOREIGN KEY constraints with ON DELETE and ON UPDATE rules (although MySQL does not support them by default). Justify in a few words any constraints (such as NOT NULL, FOREIGN KEY or UNIQUE constraints) and any update rules you use.

Create table user:

```
Create table user(
    user_id int primary key,
    nickname varchar(30) not null,
    gender varchar(10),
    region varchar(30)
);
```

Notes: this table includes all user info, which are all search-able. Every user must have nickname, so we add the not null constraint.

```
mysql> describe user;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| user_id  | int(11)     | NO   | PRI | NULL    |       |
| nickname | varchar(30) | NO   |     | NULL    |       |
| gender   | varchar(10) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

Fig1. User schema

Create table moment:

```
Create table moment(
    mom_index int primary key,
    user_id int not null,
    content varchar(3000) not null,
    post_time datetime ,
    foreign key (user_id) references user(user_id)
        on delete cascade on update cascade
);
```

Notes: all value of the foreign key attributes should not be empty, so we add not null constraints(same as the following tables). When a user is deleted from the database, then the corresponding comments should be deleted from the database either, so we add cascade delete constraint. Further, if the user update whose info, for example alter his/her

nickname, then the moment shown should also be consistent with the new nickname so we add the update cascade. Content attribute is what the user published, which must not be null. Attribute post_time is an import feature, we use it to filter the moments we care about by time range.

```
mysql> describe moment;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| mom_index | int(11)      | NO   | PRI | NULL    |       |
| user_id   | int(11)      | NO   | MUL | NULL    |       |
| content   | varchar(3000)| NO   |     | NULL    |       |
| post_time | datetime     | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
```

Fig.2 moment schema

Create table comment:

```
Create table comment(
    com_index int primary key,
    mom_index int not null,
    user_id int not null,
    content varchar(500) not null,
    comment_time datetime,
    foreign key(mom_index) references
        moment(mom_index) on delete cascade,
    foreign key(user_id) references user(user_id)
        on delete cascade on update cascade
);
```

All the foreign keys are not null, same reason as above. The mom_index doesn't need cascade update, because Wechat don't support moment update. We also need cascade delete and update towards user_id because the user info can be update or delete and these changes should affect the comment.

```
mysql> describe comment;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| com_index    | int(11)      | NO   | PRI | NULL    |       |
| mom_index    | int(11)      | NO   | MUL | NULL    |       |
| user_id      | int(11)      | NO   | MUL | NULL    |       |
| content      | varchar(500) | NO   |     | NULL    |       |
| comment_time | datetime     | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
```

Fig.3 comment schema

Create table likes:

```
Create table likes(
    like_index int primary key,
    mom_index int not null,
    user_id int not null,
    like_time datetime,
    foreign key(mom_index) references
        moment(mom_index) on delete cascade,
    foreign key(user_id) references user(user_id)
        on delete cascade on update cascade
);
```

All the foreign keys are not null, same reason as above. The mom_index doesn't need cascade update, because Wechat don't support moment update. We also need cascade delete and update towards user_id because the user info can be update or delete and these changes should affect the comment.

```
mysql> describe likes;
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| like_index | int(11)  | NO   | PRI | NULL    |       |
| mom_index  | int(11)  | NO   | MUL | NULL    |       |
| user_id    | int(11)  | NO   | MUL | NULL    |       |
| like_time  | datetime | YES  |     | NULL    |       |
+------------+----------+------+-----+---------+-------+
```

Fig.4 likes schema

b. A description of the queries you will enable over the data. The queries can be stated in SQL, but this is not required at this stage. A short and clear description of what the functionality of each query will be is sufficient at this stage.

Time-based queries:

1 At a given time range, how many moments certain users published;
2 At a given time range, how many comments certain users commented;
3 At a given time range, how many comments under a moment;
4 At a given time range, which moment is the most popular(e.g. Most commented or has most likes);
5 At a given time range, how many likes certain users liked;
6 At a given time range, how many likes a moment are liked;
7 At a given time range, which user give out most like or liked most;

Associated queries:

1 Who have commented on a given moment?
2 who have liked a given moment?
3 who have commented and liked a moment?
4 which user like most another user, e.g. Likes every moments of aother user or comments on the all the moments of another user.
Here, we plan to define a threshold t, if t >= ((comments/likes on user A)/(all comments/likes user B given)), then we infer the user A,B has strong social relationship.

Content-based queries:

These types of queries are non-traditional queries, in these queries we want search the moments or the comments users published.
1 A client can search which moments or comments are most related to a given keyword.
2 What are the similar comments or similar moments. This is very useful, since it's often whether two moments are the same one, or contains the same set of keywords.