Building Schemas

```
create table users (

    id integer auto_increment primary key,

    username varchar(255) unique not null,

    created_at timestamp default now()

);


create table photos (

    id integer auto_increment primary key,

    image_url varchar(255) not null,

    user_id integer not null,

    created_at timestamp default now(),

    foreign key(user_id) references users(id)

);


create table comments (

    id integer auto_increment primary key,

    comment_text varchar(255) not null,

    photo_id integer not null,

    user_id integer not null,

    created_at timestamp default now(),

    foreign key(photo_id) references photos(id),

    foreign key(user_id) references users(id)

);


create table likes (

    user_id integer not null,

    photo_id integer not null,
    created_at timestamp default now(),
    foreign key(user_id) references users(id),
    foreign key(photo_id) references photos(id),
    primary key(user_id, photo_id)
);

create table follows (
    follower_id integer not null,
    followee_id integer not null,
    created_at timestamp default now(),
    foreign key(follower_id) references users(id),
    foreign key(followee_id) references users(id),
    primary key(follower_id, followee_id)
);

create table tags (
  id integer auto_increment primary key,
  tag_name varchar(255) unique,
  created_at timestamp default now()
);

create table photo_tags (
    photo_id integer not null,
    tag_id integer not null,
    foreign key(photo_id) references photos(id),
    foreign key(tag_id) references tags(id),
    primary key(photo_id, tag_id)
);
```

Instagram Clone Queries (Check Views)

1. **Finding 5 oldest Users**

```
select * from users
order by created_at
limit 5;
```

2. **Weekday for most users registered**

```
select dayname(created_at) as weekday, count(*) as count from users
group by weekday order by count desc
limit 1;
```

3. **Identify Inactive Users**

```
select username from users
left join photos on photos.user_id = users.id
where image_url is null;
```

4. **Most liked Photo**

```
select username, image_url, count(*) as likes from users
left join photos on photos.user_id = users.id
join likes on likes.photo_id = photos.id
group by image_url, username order by likes desc
limit 1;
```

5. **Average user post**

```
select (select count(*) from photos) / (select count(*) from users);
```

6. **Top 5 Hashtags**

```
select tag_name, count(*) as tagged from tags

join photo_tags on photo_tags.tag_id = tags.id

group by tag_name order by tagged desc

limit 5;
```

7. **Find bots who like all photos**

```
select username, count(*) as likes from users

join likes on likes.user_id = users.id

group by username

having likes = (select count(*) from photos);
```