

PROJECT REPORT

INSY 661 – Group Project
Creating Database for a
Social Media Company

Contributors:

Marion Laniel (260431778)

Ziye Zhang (260766101)

Xitong Li (260710002)

Vaibhav Vishal (260984038)

Agathe Chapelle (260735938)

Yulin Hong (260898713)

Content:

1. Section 1
 - a. Overview
 - b. Mission
 - c. Entity-relationship diagram (ERD)
 - d. Data Dictionary
 - i. Entity information
 - ii. Attribute information
 - e. Relational Schema
2. Section 2
 - a. Data Definition Language (DDL)
 - b. Queries (objectives, assumption, query, and solution)
3. Section 3
 - Complex Query Identification
 - i. Logic
 - ii. Challenges faced
 - iii. Overall learning

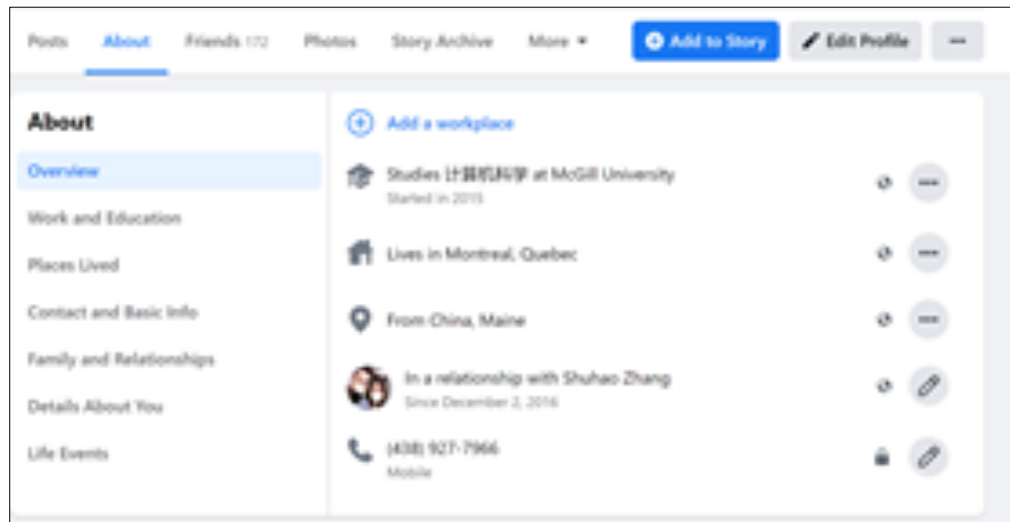
1. Section 1

a. Overview:

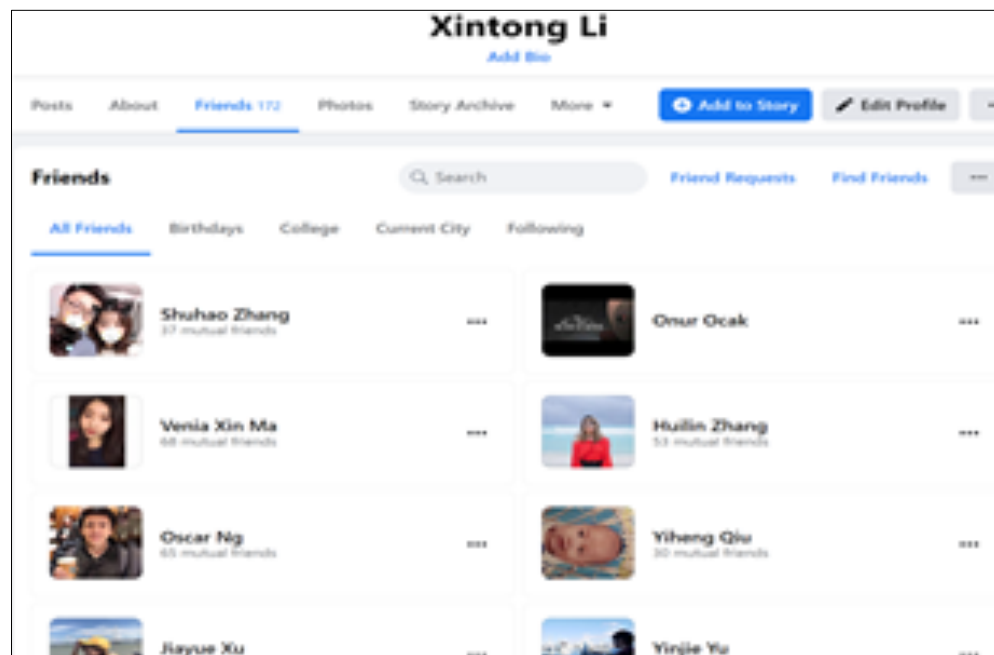
In the overviews below, we are only tracking the functionalities in the scope of our database, not the whole social media website. But we will show other advanced functionalities of the website in section three such as friend suggestion

USER:

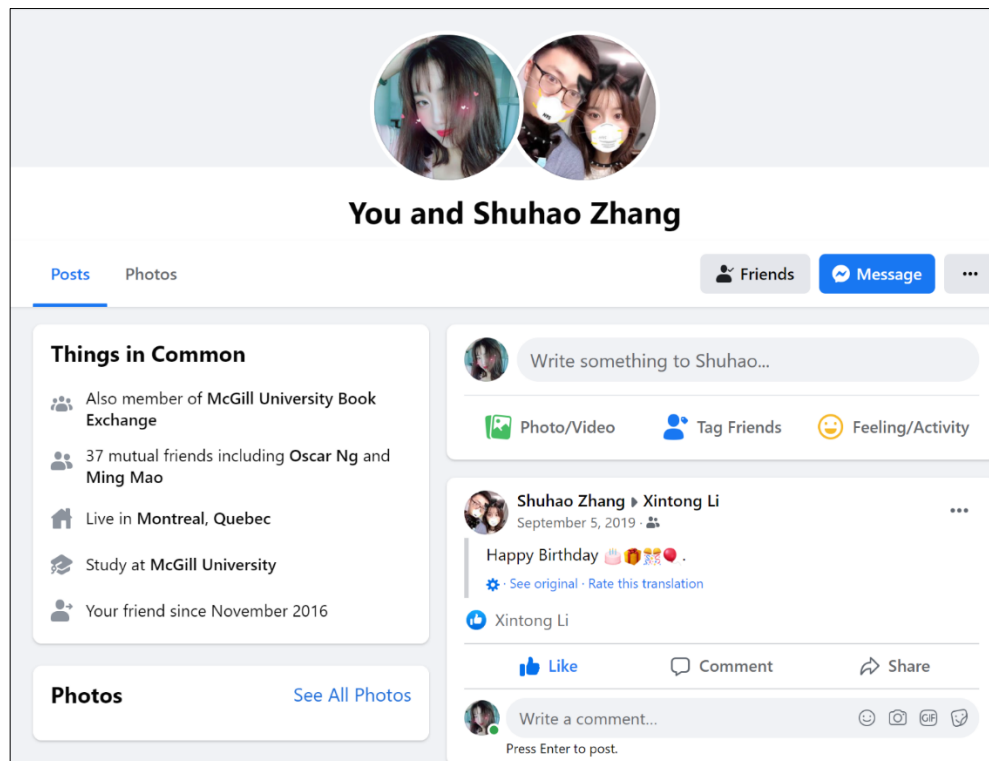
User Information



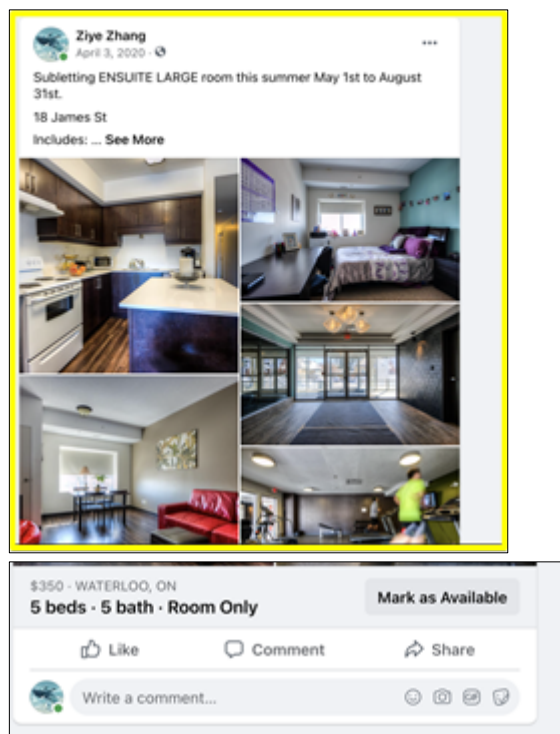
Users can have friends and our database should be able to store on which date the two users start a friendship



PROJECT REPORT



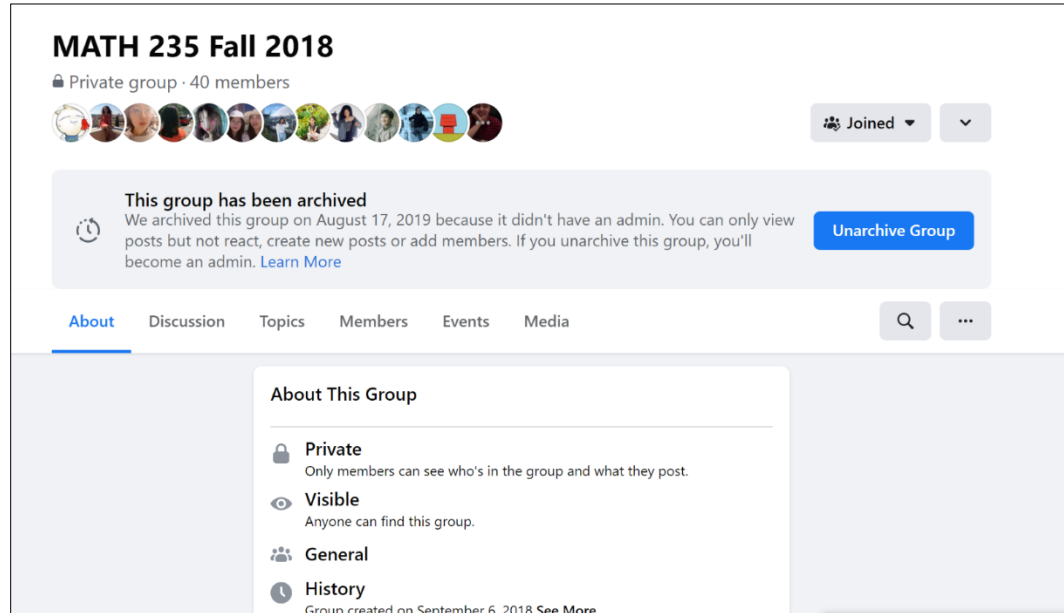
Users can post, can like, comment



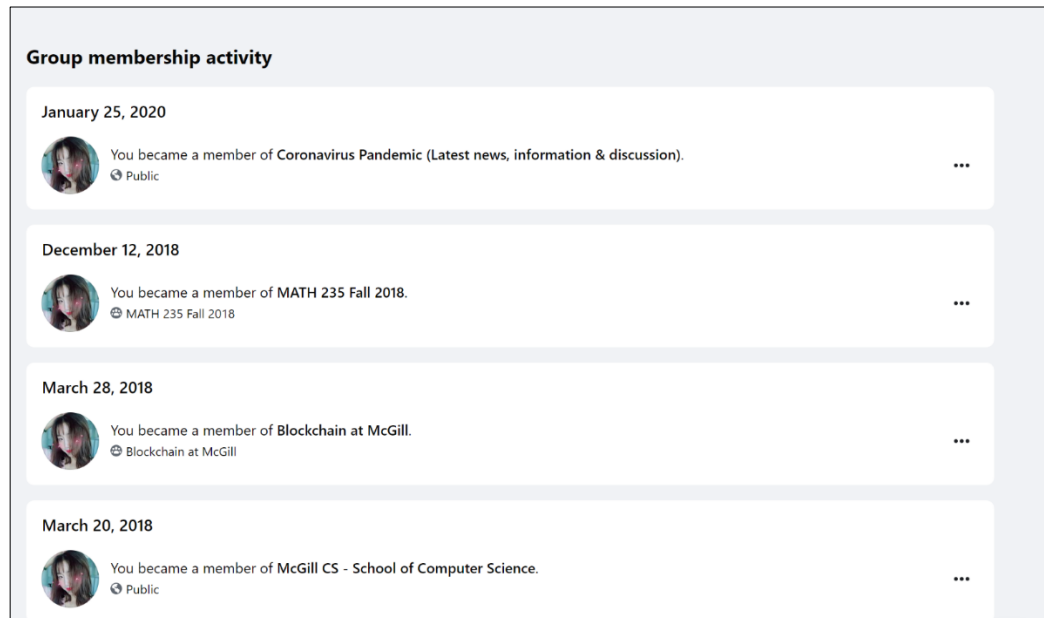
PROJECT REPORT

Groups:

Our website should be able to store group information: title, audience, description

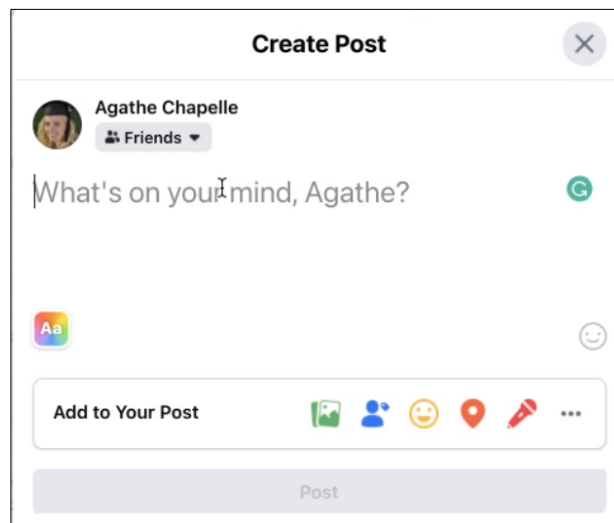


A user can join a group, and our database should be able to track on what date one user joined one group



PROJECT REPORT

Group members can post inside group discussion



Posts:

A post should contain author (which is a user), date, type, about, audience



PROJECT REPORT

Event:

Events should be able to display their basic information, Such as the title, date, price, audience, description

AUG 28 AT 8 AM EDT – AUG 29 AT 5:30 PM EDT

Legends Festival: Guildford (SELLING OUT)

Guildford, United Kingdom

About Discussion

☆ Interested

Xintong, Stay Up to Date on Coronavirus (COVID-19) Information

It's up to all of us to slow the spread of COVID-19. Everyone, including young and healthy people, should avoid large gatherings during this time. Stay up to date with public health guidelines from canada.ca.

Dismiss See Guidelines

Details

18.2K people responded

Event by **Legends Festival UK Tour - 2021, Rhythm Of The 90s** and 3 others

Guildford, United Kingdom

Tickets
www.legendsfestival.co.uk/guildford

Public · Anyone on or off Facebook

The UK's largest celebration of the world's greatest pop and rock acts is steaming into Guildford. With an incredible line up on our festival stage, everyone will love this unforge... [See More](#)

Music Kid Friendly Volunteering Musical theater

Users could show their attitudes towards an event:

☆ Interested

⌚ Going

✉ Invite

➦

⋮

PROJECT REPORT


An event can have many posts in its discussion, but one post should be only related to one event

AUG 28 AT 8 AM EDT – AUG 29 AT 5:30 PM EDT

Legends Festival: Guildford (SELLING)


Guildford, United Kingdom

About Discussion



Add a Post

PINNED POST




Legends Festival UK Tour - 2021
August 23 at 10:18 AM · 🌐

IMPORTANT PRE-EVENT INFORMATION – PLEASE READ

Thank you for purchasing tickets to The Legends Festival at Stoke Park, taking place this coming Saturday & Sunday!

Please find below some helpful information in preparation for the weekend..... [See More](#)



b. Mission

The purpose of our simulated database based on Facebook is to maintain the data that is used to support a platform where people can make friends, participate in events, join groups, and share posts and to attract advertisers to post their ads to their designated users or groups in the database.

Mission Objectives

- To perform searches on users
- To perform searches on events
- To perform searches on groups
- To perform searches on posts

- To report on users
- To report on events
- To report on groups
- To report on posts

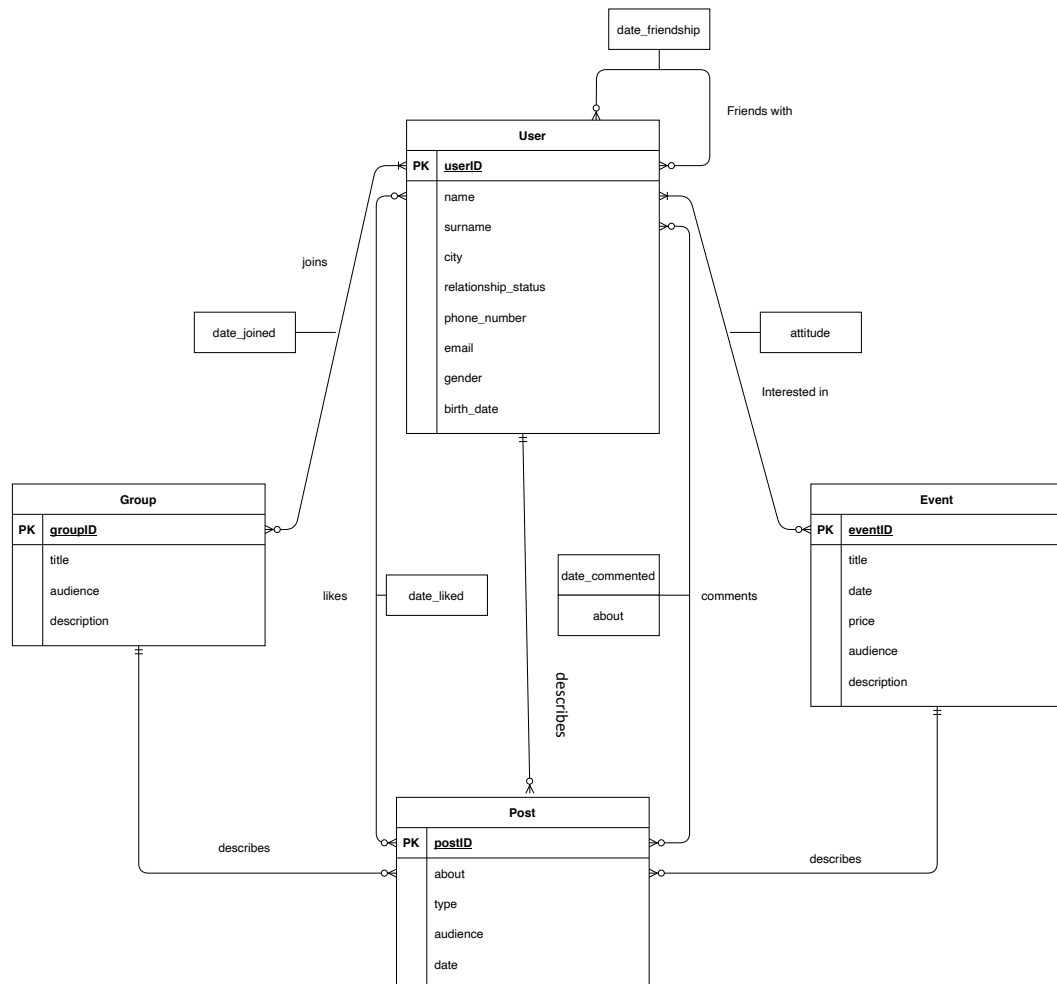
- To maintain (enter, update, and delete) data on users
- To maintain (enter, update, and delete) data on events
- To maintain (enter, update, and delete) data on groups
- To maintain (enter, update, and delete) data on posts

- To track the status of likes and comments
- To track the status of users joining groups
- To track the status of users' attitudes towards events (interested & going)
- To track the status of users' friendships

c. Entity Relationship Diagram

Assumptions:

- A user can make friends with one or many users on a certain date.
- Users can post multiple posts, comments, likes, but each post, comment, like should belong to only one user.
- We only store title, audience, description for groups.
- We only store title, date, price, audience, description for events.
- A user can join multiple groups/events, and a group/events can contain more than one user.
- Every user should have an attitude to an event.
- A user can post multiple posts inside a group/event, but each post should belong to only one group/event. A user can also post outside group/event, that is, user can also describe(i.e.post) posts.



d. Data Dictionary

i. Entity information

Entity Name	Description	Aliases	Occurrence
Users	An entity that represents the person who creates a Facebook Account.	N/A	A user can indicate his/her "City", relationship, phone number, email, gender, date of birth, name, password, etc.
Posts	An entity that links to a user. This entity usually consists of text and multimedia (not required).	N/A	A user can indicate his/her content/ type of posts, the audience who can look at the posts, the date on which it was posted, etc.
Events	An entity that lists down events that can be held by an organization or user.	N/A	A user can indicate his/her title of events, the date on which it will be, price, type of audience, and events details/description
Groups	An entity that users can join. This entity can host the event and post information.	N/A	A user can indicate his/her group details, its audience, group information, etc.
Friend	An entity that users connect with.	N/A	A user can indicate his/her list of friends and the date on which they connected
Comment	An entity that contains comments of users.	N/A	A user can indicate his/her comments and their date.
Like	An entity that contains likes done by the users.	N/A	A user can indicate his/her date of liking a post
User_Event	An entity that contains events associated with users.	N/A	A user can indicate his/her status of joining the event
User_Group	An entity that users can be part of.	N/A	A user can indicate his/her date of joining the group

ii. Attribute information

Entity Name	Attributes	Description	Data Type	Nulls	Multi-value d	Derive d	Default
Users	userID	User's unique identifier	INT(10)	No	Yes	No	None
	name	User first name	VARCHAR(255)	No		No	None
	surname	User last name	VARCHAR(255)	No		No	None
	city	User's city	VARCHAR(255)	Yes		No	None
	relationship_status	User's relationship status	VARCHAR(255)	Yes		No	Single', 'In a relationship', 'Engaged', 'Married', 'In an open relationship', 'It's complicated', 'Divorced', 'Widowed'
	phone_number	User's phone number	INT(10)	Yes		No	None
	email	User's email id	VARCHAR(255)	No		No	None
	gender	User's gender	VARCHAR(255)	Yes		No	None
	birth_date	User's Birth date	DATE	No		No	None
Posts	postID	Post unique identifier	INT(10)	No	Yes	No	Not Null
	about	Introduction of post	VARCHAR(255)	Yes		No	None
	type	Type of post	VARCHAR(255)	No		No	None
	audience	Audience having access to post	VARCHAR(255)	No		No	'Public', 'Friends', 'Specific'

PROJECT REPORT

							Friends', 'Only me'
	date	Creation date of post	DATE	No		No	None
Events	eventID	Unique event identifier	INT(10)	No	Yes	No	Not Null
	title	Events title	VARCHAR(255)	No		No	None
	date	Events date	DATE	No		No	None
	price	Events ticket/entry price	DECIMAL(10,2)	Yes		No	None
	audience	Type of audience	VARCHAR(255)	No		No	'Private', 'Public', 'Friends'
	description	Events summary	VARCHAR(255)	Yes		No	None
Groups	groupID	Unique group identifier	INT(10)	No	Yes	No	Not Null
	title	Group's name/title	VARCHAR(255)	No		No	None
	audience	Group members/audience	VARCHAR(255)	No		No	'Public', 'Private'
	description	About group	VARCHAR(255)	Yes		No	None
Friend	date_friendship	Date of friendship	DATE	No	No	No	None
Comment	about	Content of comment	VARCHAR(255)	Yes	Yes	No	None
	date_commented	Date of commenting	DATE	No		No	None
Like	date_liked	Date of liking	DATE	No	No	No	None
User_Event	attitude	going or not or maybe	VARCHAR(255)	No	No	No	'Going', 'Maybe', 'Can't go'
User_Group	date_joined	Date user joined the group	DATE	No	No	No	None

e. Relational Schema

EVENT (EVENTID, TITLE, DATE, PRICE, AUDIENCE, DESCRIPTION)

GROUP (GROUPID, TITLE, AUDIENCE, DESCRIPTION)

USER (USERID, NAME, SURNAME, CITY, RELATIONSHIP_STATUS,
PHONE_NUMBER, EMAIL, GENDER, BIRTH_DATE)

POST (POSTID, USERID, GROUPID, EVENTID, ABOUT, TYPE, AUDIENCE, DATE)

COMMENT (POSTID, USERID, ABOUT, DATE_COMMENTED)

LIKE (USERID, POSTID, DATE_LIKED)

FRIEND (USERID, FRIENDID, DATE_FRIENDSHIP)

USER_GROUP (GROUPID, USERID, DATE_JOINED)

USER_EVENT (EVENTID, USERID, ATTITUDE)

2. Section 2

a. DDL and DML

Please refer to the SQL script submitted along with this report.

b. Queries (objectives, assumption, query, and solution)

Query: 1

Objective: Figure out if relationship (marital status) affects the number of posts posted by people.

Code:

```
SELECT
    relationship_status AS Status,
    COUNT(postID) / COUNT(DISTINCT (relationship_status)) AS 'post per user'
FROM
    (SELECT
        userID, name, relationship_status
    FROM
        User
    WHERE
        relationship_status IS NOT NULL) u
JOIN
```

PROJECT REPORT

Post ON post.userID = u.userID
GROUP BY relationship_status
ORDER BY 'post per user' DESC;

Output Screenshot:

Status	Post per User
Engaged	1.0000
Married	6.0000
Single	2.0000
Widowed	1.0000

Query: 2

Objective: Figure out if government employees are less likely to comment and like posts due to their job rigidity.

Assumptions: Government people are those whose email ends with .gov

Code:

```
SELECT
    'Government Official',
    COUNT(postID) / COUNT(DISTINCT (userID)) AS 'Avg num of likes&comments'
FROM
    (SELECT
        reaction.userID, reaction.postID
    FROM
        User
    JOIN (SELECT
        userID, postID, date_liked AS date_reaction
    FROM
        `Like` UNION SELECT
        userID, postID, date_commented
    FROM
        `Comment`) reaction ON User.userID = reaction.userID
    WHERE
        email LIKE '%gov') temp
UNION SELECT
    'Non Government Official',
    COUNT(postID) / COUNT(DISTINCT (userID)) AS 'Avg num of likes&comments'
FROM
    (SELECT
        reaction.userID, reaction.postID
    FROM
        User
    JOIN (SELECT
        userID, postID, date_liked AS date_reaction
```

PROJECT REPORT

```
FROM
  `Like` UNION SELECT
    userID, postID, date_commented
FROM
  `Comment`) reaction ON User.userID = reaction.userID
WHERE
  email NOT LIKE '%gov') temp;
```

Output Screenshot:

	Government Official	Avg num of likes&comments
▶	Government Official	3.0000
▶	Non Government Official	2.1176

Query: 3

Objective: Find out if the price of an event affects people's attitude towards it.

Assumptions: Consider 'Going' as a positive attitude. The others are negative.

Code:

```
SELECT
  x.level Level,
  x.total_attitude 'Total Attitudes',
  y.num_Going 'Number of Going',
  CONCAT(FORMAT(y.num_Going / x.total_attitude * 100,
    2),
    '%') AS '% of Going'
FROM
  (SELECT
    level, COUNT(*) AS total_attitude
  FROM
    (SELECT
      `Event`.eventId,
      User_event.userID,
      User_event.attitude,
      CASE
        WHEN price > 100000 THEN 'Extremely High'
        WHEN price > 200 THEN 'High'
        WHEN price > 0 THEN 'Low'
        ELSE 'Free'
      END `level`
    FROM
      `Event`
```


PROJECT REPORT

```

JOIN User_Event ON `Event`.eventID = User_event.eventID) temp1
GROUP BY level) x
JOIN
(SELECT
  level, COUNT(attitude) AS num_Going
FROM
  (SELECT
    `Event`.eventID,
    User_event.userID,
    User_event.attitude,
    CASE
      WHEN price > 100000 THEN 'Extremely High'
      WHEN price > 200 THEN 'High'
      WHEN price > 0 THEN 'Low'
      ELSE 'Free'
    END `level`
  FROM
    `Event`
  JOIN User_Event ON `Event`.eventID = User_event.eventID) temp2
WHERE
  attitude LIKE 'Going'
GROUP BY level) y ON x.level = y.level
ORDER BY y.num_Going / x.total_attitude DESC;

```

Output Screenshot:

	Level	Total Attitudes	Number of Going	% of Going	
▶	Free	3	3	100.00%	
▢	Extremely High	2	1	50.00%	
	Low	6	2	33.33%	
▢	High	11	3	27.27%	

Query: 4

Objective: Find the most active person on the list

Assumptions: The most active person is the one who has the maximum number of events attended, groups joined, and posts posted. That is, Activities = # of groups+# of events+# of posts.

Code:

```

SELECT
  name Name, Activities
FROM
  (SELECT
    u.userID,

```

PROJECT REPORT

```

        u.name,
        a.num_of_group,
        b.num_of_event,
        c.num_of_post,
        a.num_of_group + b.num_of_event + c.num_of_post AS Activities
FROM
    User u
LEFT JOIN (SELECT
    userID, COUNT(groupID) num_of_group
FROM
    User_Group
GROUP BY userID) a ON u.userID = a.userID
LEFT JOIN (SELECT
    userID, COUNT(EventID) num_of_event
FROM
    User_Event
GROUP BY userID) b ON u.userID = b.userID
LEFT JOIN (SELECT
    userID, COUNT(postID) num_of_post
FROM
    Post
WHERE
    userID IS NOT NULL
GROUP BY userID) c ON u.userID = c.userID) d
ORDER BY Activities DESC
LIMIT 1;

```

Output Screenshot:

	Name	Activities
▶	Dorine	8

Query: 5

Objective: Select people who attend expensive events so that Facebook can target those users and places luxury products' advertisements on their homepages

Assumptions: Assume users with the attitude 'going' are attending these events.

Code:

```

SELECT
    User_Event.userID,
    price

```

PROJECT REPORT

```

from Event
join
User_Event
on Event.eventID = User_Event.eventID
where attitude = 'going'
order by price desc
limit 3;

```

Output Screenshot:

userID	price	
5	57179940.62	
3	926.68	
9	282.00	

Query: 6

Objective: Suggest friends for the user named Goldie. The suggestion is based on the events Goldie attended. People who attended the same events as Goldie but are not friend with Goldie will be recommended.

Code:

```

select
concat(User. name, ' ', User. surname) as 'suggested friends for attending the same
event with Goldie'
from
User_Event
join
User
on User_Event.userID = User.userID
where
User_Event.eventID in (select
eventID
from User_Event
where userID in (select userID
from User
where name = 'Goldie'))
and
User_Event.userID not in (select
friendID
from friend
where userID in (select userID
from User

```

where name = 'Goldie'));

Output Screenshot:

suggested friends for attend same event with Goldie	
▶ Goldie Spurr	
Cassidy Patel	
Result 63	

Query: 7

Objective:

Suggest friends for the user named Jacquenetta based on her friends. Her friends' friends will be recommended if they are not friends with Jacquenetta.

Code:

```
select
concat(User.name, ' ', User.surname) as name,
User.city,
friends.f_name as is_friend_with
from User
join(
select
distinct friendID as ID,
friend.userID as f
From friend
where
friend.userID
in (
select
friendID as userID
from friend
where userID in
(select userID
from User
where name = 'Jacquenetta'))
) as suggest_friends
on User.userID = suggest_friends.ID
join
(select
concat(User.name, ' ', User.surname) as f_name,
User.userID as ID
from
user) as friends
on suggest_friends.f = friends.ID
;
```

PROJECT REPORT

Output Screenshot:

	name	city	is_friend_with	
▶	Everard Morforth	Pilar	Dorine Byrnes	
	Dido Lorey	Lampa	Dorine Byrnes	
	Dido Lorey	Lampa	Goldie Spurr	
	Juliet Roach	Sheikhupura	Goldie Spurr	

Query: 8

Objective:

Suggest groups user Jacquenetta might be interested in, based on her friends' interests.
Order by the number of Jacquenetta's friends in that group.

Code:

```
select
  User_Group.groupID,
  `Group`.title,
  count(User_Group.userID) as 'number of friends in this group'
from User_Group
join `Group`
  on User_Group.groupID = `Group`.groupID
where User_Group.userID in(
  select
    friendID
  From Friend
  where Friend.userID in
    (select userID
     from User
     where name = 'Jacquenetta'))
AND
  User_Group.groupID not in(
  select
    User_Group.groupID
  From User_Group
  where User_Group.userID in
    (select userID
     from User
     where name = 'Jacquenetta'))
group by groupID
order by count(User_Group.userID) desc;
```

Output Screenshot:

PROJECT REPORT

groupID	title	number of friends in this group
3	Politics Department	2
6	ORGB660	1
8	INSY661	1
Result 68		

Query 9

Objective: Select the event and show its participants, if anyone is going to it. By doing so, we can analyze those events that tend to attract customers

Code:

```
select event.eventID, event.title, user.name, user.surname
```

```
from user_event
```

```
inner join event on user_event.eventID = event.eventID
```

```
inner join user on user_event.userID = user.userID
```

```
where user_event.attitude = 'Going';
```

Output Screenshot:

	eventID	title	name	surname
▶	1	Ski Trip	Jacquenetta	Servis
	11	Museum	Jacquenetta	Servis
	3	Celine Dion Concert	Jacquenetta	Servis
	11	Museum	Jacquenetta	Servis
	11	Museum	Jacquenetta	Servis
	4	Daytrip to the mall	Jacquenetta	Servis
	7	Cooking Class	Jacquenetta	Servis
	2	Graduation 2022	Jacquenetta	Servis
	5	Party at my house	Jacquenetta	Servis
	1	Ski Trip	Dorine	Byrnes
	11	Museum	Dorine	Byrnes
	3	Celine Dion Concert	Dorine	Byrnes

PROJECT REPORT

Query: 10

Objective: Suggest groups Dido might be interested in, we knew that Dido has joined the group “McGill”, we could suggest all the other groups contain the keyword “McGill” based on this.

Code:

```
select * from `group`  
  
where title like (  
concat("%", (select title from (  
select Dido.name, `group`.title from user_group  
inner join(  
select * from user  
where name = 'Dido'  
)  
as Dido  
on user_group.userID = Dido.userID  
inner join `group` on `group`.groupID = user_group.groupID  
)  
as a), "%")  
)  
AND TITLE <> (  
select title from (  
select Dido.name, `group`.title from user_group  
inner join(  
select * from user  
where name = 'Dido'  
)  
as Dido  
on user_group.userID = Dido.userID  
inner join `group` on `group`.groupID = user_group.groupID  
)  
as a  
);
```

Output Screenshot:

PROJECT REPORT

	groupID	title	audience	description
▶	1	MMA McGill	Private	2021 cohort
	12	McGill Math	Private	All burnsidars!
	13	McGill Book Exchange	public	a place to buy/sell cheap textbooks!

Query: 11

Objective: Find the author of the post which is commented by most users, display the post_ID and the number of comments it received as well.

Code:

```
select post.postID as mostPopularPost, user.userID as mostPopularAuthor,
c.NumberOfComments from post
inner join (
select * from (
select count(distinct(userID)) as NumberOfComments, postID
from `Comment`
group by postID
)as a
where a.NumberOfComments = (
select max(NumberOfComments) from (
select count(userID) as NumberOfComments, postID
from `Comment`
group by postID
)as b
)
)as c
on post.postID = c.postID
inner join user on post.userID = user.userID
```

Output Screenshot:

	mostPopularPost	mostPopularAuthor	NumberOfComments
▶	1	1	5

Query: 12

Objective: count the number of potential attendees to an upcoming event

Assumption: Potential attendees include “maybe” and “not going”

Code:

```
Select Event.eventID, Event.title, A.NumberOfAttendees
```


PROJECT REPORT

From Event

Join (Select count(User_Event.userID) NumberOfAttendees, User_Event.eventID

From User_Event

Where User_Event.attitude != "Can't go"

Group by eventID) as A on Event.eventID=A.eventID

Output:

	eventID	title	NumberOfAttendees
▶	1	Ski Trip	2
	2	Graduation 2022	2
	3	Celine Dion Concert	3
	4	Daytrip to the mall	1
	5	Party at my house	3
	7	Cooking Class	3
	9	MMA Graduation	1
	10	Animal Shelters Volunteer	2
	11	Museum	3

Query: 13

Objective: Determine the average age of a Facebook user

Code:

Select Avg(B.Age) as AverageAge

From (Select ("2021"- A.BirthYear) as Age

From (Select extract(Year from birth_date) as BirthYear

From `User`) as A) as B

Output:

	AverageAge
▶	37.45

Query: 14

Objective: Find the users who have more male friends than females.

Code:

SELECT

user.name, b.userID, b.Male_friend, c.Female_friend

FROM

(SELECT

userID, COUNT(gender) Male_friend

FROM

(SELECT

friend.userID, friend.friendID, user.gender

```

FROM
  Friend
JOIN user ON user.userID = Friend.friendID) a
WHERE
  gender = 'Male'
GROUP BY userID) b
LEFT JOIN
(SELECT
  userID, COUNT(gender) Female_friend
FROM
  (SELECT
    friend.userID, friend.friendID, user.gender
  FROM
    Friend
  JOIN user ON user.userID = Friend.friendID) a
  WHERE
    gender = 'Female'
  GROUP BY userID) c ON b.userID = c.userID
  JOIN
  user ON b.userID = user.userID
WHERE
  b.Male_friend > c.Female_friend;

```

Output Screenshot:

	name	userID	Male_friend	Female_friend
▶	Jacquenetta	1	6	1
▶	Everard	3	2	1

Query 15

Objective: Suggest a friend for the user named Jacquenetta, based on strangers who commented on Jacquenetta's post.

Assumption: a person who is not friends with Jacquenetta is considered to be a stranger.

Code:

```

select
  concat(User.name, ' ', User.surname) as 'suggested friend',
  Comment.date_commented
from
  Comment
join User
on Comment.userID = User.userID
where

```

```

Comment.postID in
(select
Post.postID
from
Post
where Post.userID in
(select User.userID
from User
where name = 'Jacquenetta'))
AND
Comment.userID NOT IN
(select
Friend.friendID
from
Friend
where Friend.userID in
(select User.userID
from User
where name = 'Jacquenetta'))
AND
Comment.userID NOT IN
(select User.userID
from User
where name = 'Jacquenetta')

```

Output Screenshot:

Result Grid		Filter Rows:
suggested friend	date_commented	
▶ Everard Morforth	2016-02-25	
Amethyst Kline	2024-02-26	
Melanie Clemons	2018-08-29	

Query 16

Objective: identifying the upcoming event in the next month

Code:

```

select eventID
from `Event`
where extract(month from `date`) = (select (extract(month from sysdate())) + 1
as next_month)
and extract(year from `date`) = (select (extract(year from sysdate())))

```

Output:

PROJECT REPORT

eventID
6

Query 17

Objective: identifying users who are engaging the most with posts by evaluating their number of likes attributed to different posts.

Code:

```
select userID, count(distinct postID) as nb_likes  
from `Like`  
group by userID  
order by userID
```

Output:

userID	nb_likes
1	2
3	1
4	1
5	1

Query 18

Objective: determining the number of friends per user.

Code:

```
select userID, count(distinct friendID) as nb_friends  
from Friend  
group by userID  
order by userID
```

Output:

userID	nb_friends
1	2
2	2
5	1

Query 19

Objective: determining the proportion of posts that contain a photo attachment.

Code:

```
select t.nb_photos/s.nb_posts as proportion_posts_photos  
from(
```

PROJECT REPORT

```
select count(distinct postID) as nb_posts
from Post) as s,
(
select count(distinct postID) as nb_photos
from Post
where type like 'Photo') as t
```

Output:

proportion_posts_photos
0.6000

Query 20

Analyze the influence of marital status on people's consumption perceptions through our social media platform

Code:

```
select Count(Marital_Status)/(
select count(*) as total_people_going from (
select Marital_status, userID, eventID,
CASE
```

```
    WHEN price > 900 THEN 'High'
    ELSE 'Low'
```

```
END `Price_level`,
```

```
CASE
```

```
    when attitude = "Can't go" then "not interested"
    else "interested"
```

```
end "attitude_level"
```

```
from (
```

```
select user_event.userID, user_event.eventID, u.Marital_Status, user_event.attitude,
event.price
```

```
from user_event
```

```
inner join
```

```
(
```

```
SELECT
```

```
    userID,
    gender,
```

```
CASE
```

```
    WHEN relationship_status = 'Married' THEN 'Married'
```

```

        ELSE 'Unmarried'

    END `Marital_Status`

FROM user
)as u
on u.userID = user_event.userID
inner join event on user_event.eventID = event.eventID

)as a
)as b
where attitude_level <> 'not interested'

) as percent_married_people_going_to_expensive_event
from (
select * from (
select Marital_status, userID, eventID,
CASE

        WHEN price > 900 THEN 'High'
        ELSE 'Low'

    END `Price_level`,

CASE
    when attitude = "Can't go" then "not interested"
    else "interested"
end "attitude_level"

from (
select user_event.userID, user_event.eventID, u.Marital_Status, user_event.attitude,
event.price
from user_event
inner join
(
SELECT
    userID,
    gender,

    CASE

        WHEN relationship_status = 'Married' THEN 'Married'

        ELSE 'Unmarried'
    
```

```
END `Marital_Status`
```

```
FROM user
)as u
on u.userID = user_event.userID
inner join event on user_event.eventID = event.eventID

)as a
)as b
where attitude_level <> 'not interested'
)as c
where Marital_Status = 'Married'
and price_level = 'High';
```

Output:

	percent_married_people_going_to_expensive_event
▶	0.0476

3. Section 3

Complex / Interesting Query Identification

Two interesting queries: Query 3 and Query 10

Query 3

- i. Idea: When analyzing users' behaviors towards events, we think the price of an event is one of the factors that affect people's attitude (i.e., one of the options among "Going", "Maybe", "Can't go"). Therefore, it would be useful to query the relationship between price levels and percentage of Going, where $\%Going = \#_of_Going / (\#_of_Going + \#_of_Maybe + \#_of_Can'tGo)$
- ii. Logic: Since it is harder for SQL language to analyze continuous price data than using other programming languages where we can run linear regressions at ease such as Python and R, we divide the continuous price data into four levels using Case functions, which are "Extremely High", "High", "Low", and "Free". We first query and count the total number of attitudes towards each event, then count the number of going towards each event, finally we inner join these two tables based on the 4 price levels and derive the percentage of going in descending order.
- iii. Challenges faced

PROJECT REPORT

1. People who do not go may fully ignore the event posts and do not express their attitudes on the platform. This may decrease the number of "Can't go" and thus the % of going may falsely seem higher
2. As mentioned in logic, linear regression might be more useful for the continuous data unless there are strong patterns for clusters
3. Data is not enough, especially for 'extremely high'. This may cause the % of Going to seem high in the result
4. The two counts cannot be done using a single table because the latter needs additional conditions.

iv. Overall learning:

1. The MySQL built-in function Case function can be helpful for grouping continuous data into clusters when the cutoff values are determined
2. MySQL is a DBMS and SQL language is a query language. People should not be constrained to SQL when they work in data analysis. Data can always be loaded to other platforms for further exploration

Query 10

i. Idea: When browsing the Facebook site, we found that Facebook uses text-matching to suggest groups that users might like, so we wanted to implement this feature on our social platform as well.

ii. Logic: We first find the groups that the target user has joined, then use MYSQL's LIKE statement and regular expression to find other groups with similar names, and don't forget to exclude the groups that the target user has joined in the final recommendation.

iii. Challenges: While splitting a string in python is easy, it's very difficult to do in MySQL, so we can only suggest groups for users that contain the name of their existing group, and it's very difficult to cut out a part of the existing group name for recommendation

iv. Learnings: When using subquery inside the MYSQL like statement, we must remember to add "%" to the concat statement, otherwise the subquery return will not be used properly, and we will not get the desired output. (for example, LIKE (CONCAT("%", (...), "%")))

One complex query: Query 20

Query 20

PROJECT REPORT

- i. Idea: People are likely to become frugal when they get married because most of them have to think about raising children, so we wanted to examine this trend through our social platforms.
- ii. Logic: we first join user_event with user and event table, from these we only need marital status, userID, eventID, attitude, price. Then we select out those records with high price, and count how many of them are married, finally use the previous number we got to divide by the whole number.
- iii. Challenges: When group by, not only marital status needs to be taken into account, but also price, that is, the impact of both on people, which is somewhat tricky here. Because only one column can be grouped by when group by.
- v. Learnings: When faced with the tricky problem of having two or even more columns that need to be grouped by, we can instead consider representing the result in probabilistic form, so that we can reduce the number of columns that need to be grouped by. And it does not affect our analysis, because after deriving one probability, we can subtract it from 1 to get another probability, and this comparison also gives us a sense of who has more influence on the result.

PROJECT REPORT

Appendix #1: Draft ERD – first attempt at developing the ERD. We modified this ERD in order to accurately capture the many to many relationships.

