INSY 661-075

Database and Distributed Systems for Analytics

Presented to Professor Animesh Animesh

Group Project – Bumble Database

Prepared by Group 8:

Alexis de Pampelonne 260734005

Cristina Esposito 260744222

Yashica Na 260945954

Yichen Wang 260761601

Diwei Zhu 260761307

Tuesday, September 14th, 2021

Contents

Section 1 – Bumble Database 1

Overview of the Business Scenario 1

Mission 2

Objective 2

Studying the App and its Competitors 3

USER FLOW 4

ERD 6

Data Dictionary 7

Description of Attributes 8

Relational Schema 12

Logical Model 12

Physical Model 14

Section 2 – Queries 15

Theme 1- App features 15

Query 1 15

Theme 2 – Feature exploration 16

Query 2 16

Query 3 18

Query 4 19

Query 6 23

Query 7 24

Theme 3 – Matching / Combination Metrics 26

Query 8 26

Query 9 26

Theme 4 – Performance Metrics 27

Query 10 27

Query 11 28

Query 12 30

Query 13 31

Query 14 32

Theme 5 – User Targeting 32

Query 15 32

Query 16.. 33
Query 17.. 35
Section 3 – Most Interesting and Complex Queries.. 37
Query 18.. 37
Query 19.. 41
Query 20 - New feature.. 44

Section 1 – Bumble Database

Overview of the Business Scenario

We have decided to build a dating app database inspired by the model of Bumble. Bumble is a free dating app that rapidly got popular thanks to its "girls make the first step" approach (female users are the ones that initiate a discussion). Bumble alike the various other dating apps offers the basic functionalities of a dating app:

- **Connecting people** with complementary profiles, based on various features
    - Gender
    - Sexual orientation
    - Location
    - Hobbies
    - Lifestyle (smoking, drinking…)
    - Opinions (religion, politics…)
- Secondly, Bumble has a **subscription-based model**, meaning users can subscribe to different offers to access additional features:
    - unlimited number of swipes per day
    - cancelling swipes
    - and more…
- We decided to add that **subscription approach** in our model, as well as develop further functionalities:
    - Premium allows you to manually change your location
    - **The possibility to host and attend events:**
        - Indeed, few dating apps today offer to their users tailored events to their needs: dating, making friends etc.
        - By understanding our users, we aim at offering a new way of connecting with each other, away from your phone.
- Lastly, we explored how our model enables us to create value from the data, to perform:
    - **User targeting / user analytics:**
        - Understand the demographics of our customer base and tailor more personalised experiences.
        - Find trends in our data: e.g., popular hobbies in a segment of our users
        - Sponsor events or users to the appropriate audience
    - **Financial metrics**
        - Find conversion rate of our subscription model
    - **Performance** metrics
        - Evaluate the algorithm's suggestion capability (the average amount of swipes per month): this metric is vital to understand if our dating app is useful: if a user must perform a lot of swipes, then he doesn't meet "the one" quickly enough.
        - Evaluate our premium plan's efficiency:
            - Do our premium members get higher number of matches?

Mission

Currently, the purpose of the Bumble database system is to record users' data and generate combinations to support the main app's function: match making people based on various interests and preferences.

Parallelly, Bumble's database is used as well to generate user reports and enable user targeting for brands willing to market products to its users. Such brands can use our reports to better understand demographics of their industry: e.g., with our database we can generate reports exploring the demographics of certain hobbies (e.g., cooking) in a specific location (e.g., San Francisco) enabling brands related to that hobby (e.g., Tefal) to build better campaigns.

Objective

- To maintain (enter, update, and delete) data on user logins
- To maintain (enter, update, and delete) data on user profiles
- To maintain (enter, update, and delete) data on users' subscriptions
- To maintain (enter, update, and delete) data on events

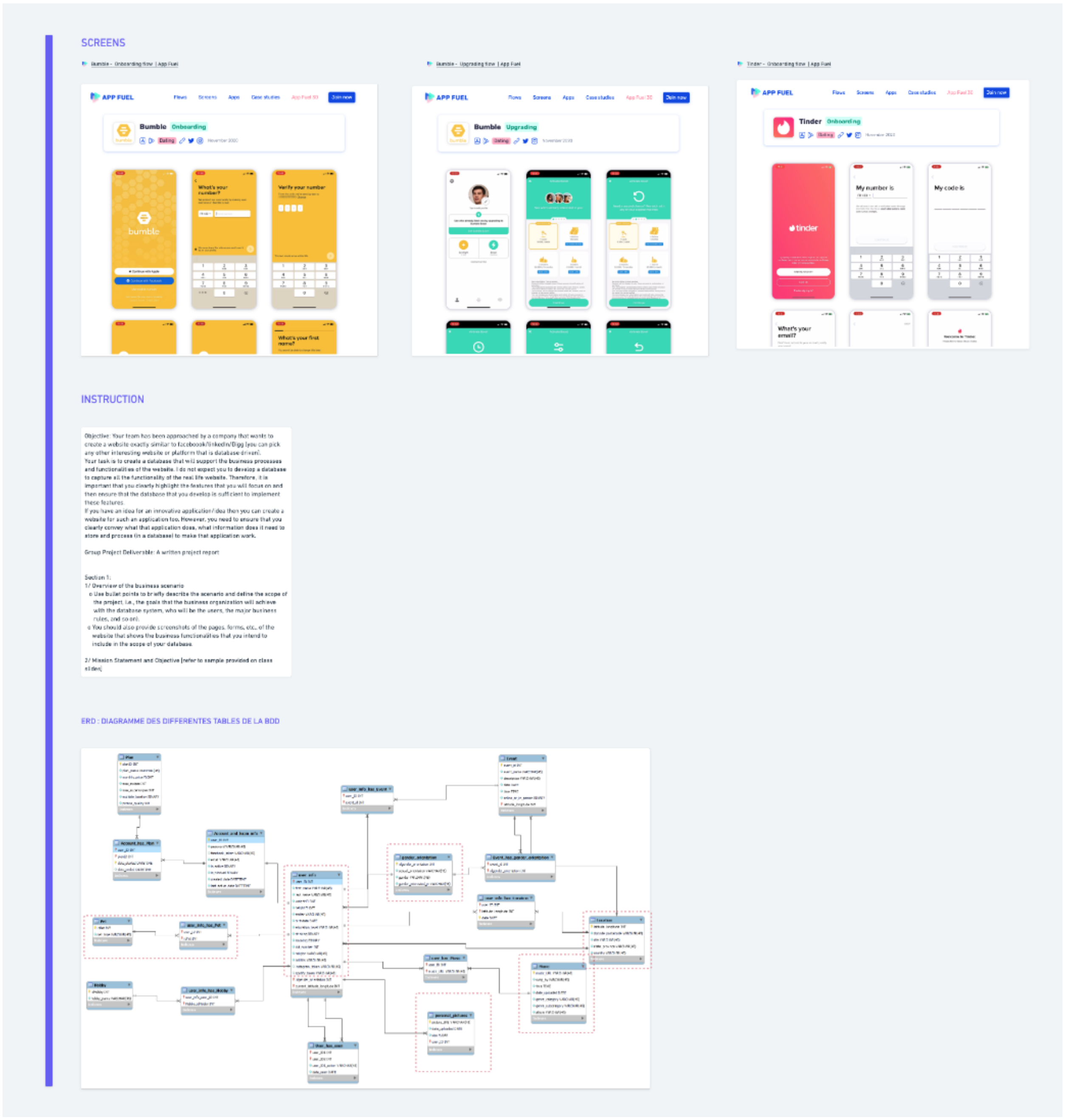
- To perform matching combinations on users
- To perform matching combinations between events and users
- To perform searches on users matched
- To perform searches on events in a user's area
- To perform

- To track user's location and update the suggested users based on new location
- To track the status of ongoing matches
  - user1 swiped right / interested in → user2
  - did user2 swipe right?
- To track user's usage on the app
  - Referring to financial and performance metrics

- To report on user metrics: usage, payments, and profiles…
- To report on event metrics: attendees, location, target users…

Studying the App and its Competitors

To truly grasp how Bumble's database was structured, we had to immerse ourselves into the app's user experience, as well as what was being done by its competitors. Via the study of the various screen flows, we then decided to create our own app's user flow.


SCREENS
Bumble - Onboarding flow | App Fuel
Bumble - Upgrading flow | App Fuel
Tinder - Onboarding flow | App Fuel
INSTRUCTION
ERD : DIAGRAMME DES DIFFERENTES TABLES DE LA BDD

Objective: Your team has been approached by a company that wants to create a website exactly similar to facebook/linkedin/Digg (you can pick any other interesting website or platform that is database driven).
Your task is to create a database that will support the business processes and functionalities of the website. I do not expect you to develop a database to capture all the functionality of the real life website. Therefore, it is important that you clearly highlight the features that you will focus on and then ensure that the database that you develop is sufficient to implement these features.
If you have an idea for an innovative application/idea then you can create a website for such an application too. However, you need to ensure that you clearly convey what that application does, what information does it need to store and process (in a database) to make that application work.

Group Project Deliverable: A written project report

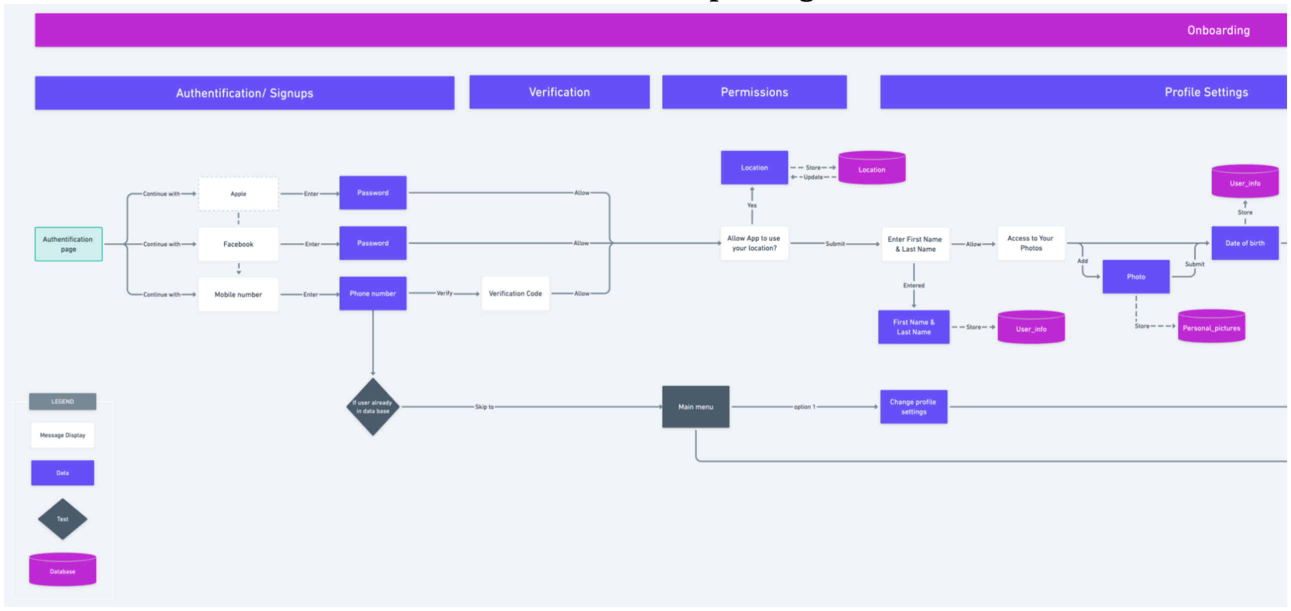
Section 1:
1/ Overview of the business scenario
- Use bullet points to briefly describe the scenario and define the scope of the project, i.e., the goals that the business organization will achieve with the database system, who will be the users, the major business rules, and so on).
- You should also provide screenshots of the pages, forms, etc., of the website that shows the business functionalities that you intend to include in the scope of your database.

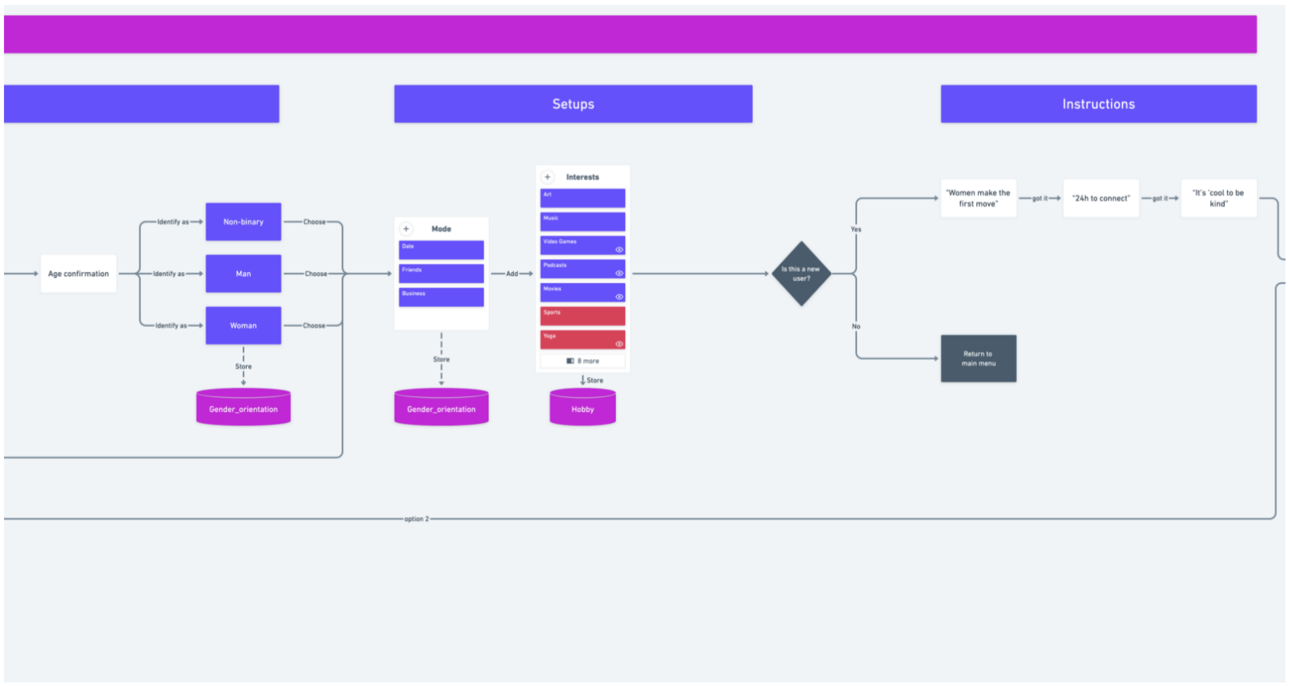
2/ Mission Statement and Objective (refer to sample provided on class slides)

USER FLOW

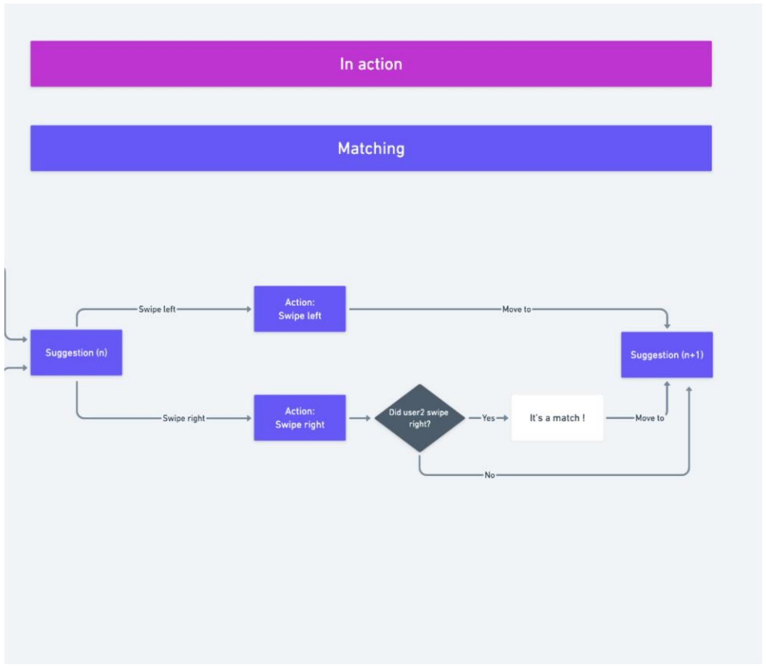
Based on Bumble's user flow, we constructed a user flow chart to understand how our data was going to navigate through the app's various basic functionalities: creating / updating a profile and swiping right or left on suggested users.

Profile Creation / Updating

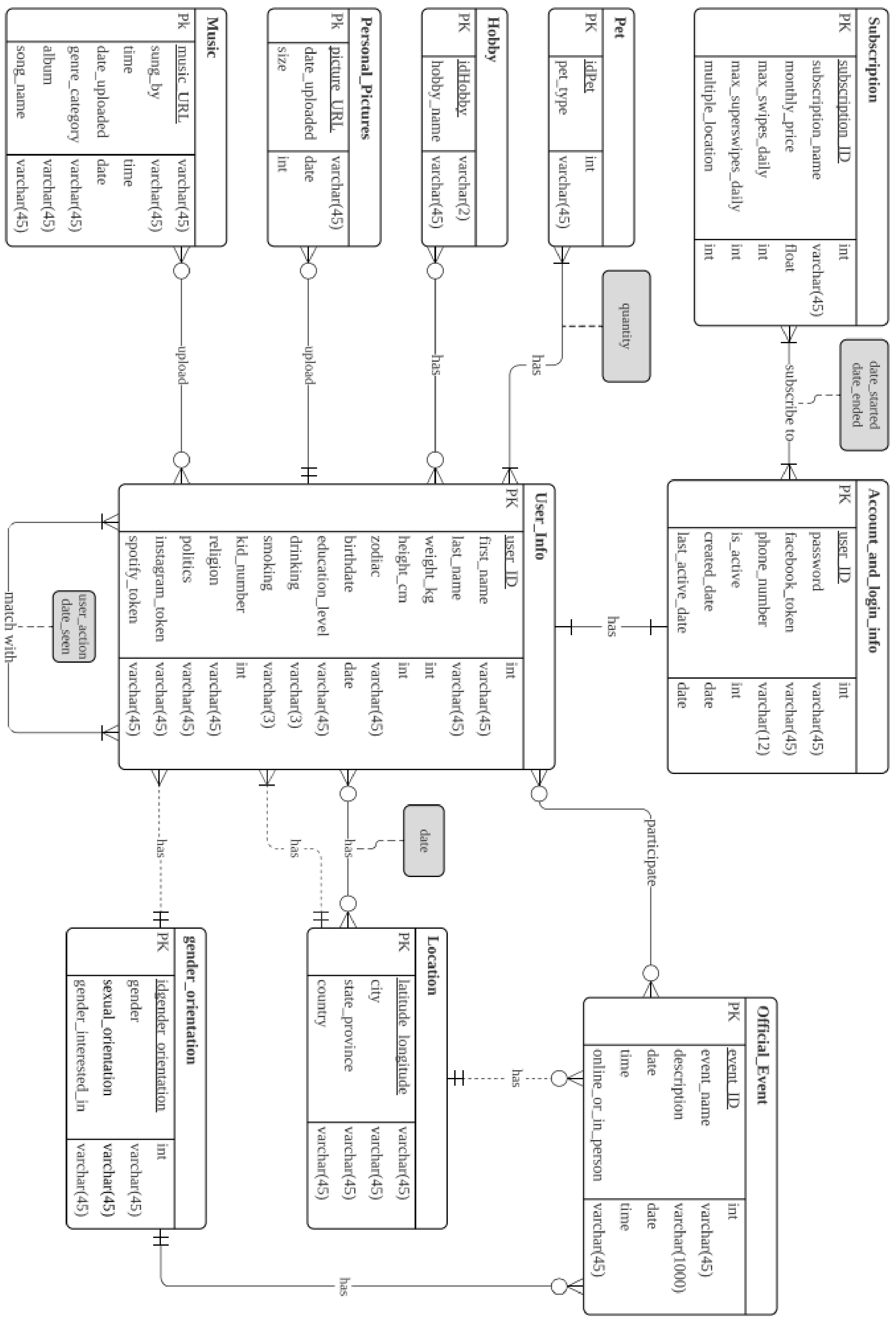




User Matching



ERD


Subscription
PK subscription_ID int
subscription_name varchar(45)
monthly_price float
max_swipes_daily int
max_superswipes_daily int
multiple_location int
Pet
PK idPet int
pet_type varchar(45)
Hobby
PK idHobby varchar(2)
hobby_name varchar(45)
Personal_Pictures
Pk picture_URL varchar(45)
date_uploaded date
size int
Music
Pk music_URL varchar(45)
sung_by varchar(45)
time time
date_uploaded date
genre_category varchar(45)
album varchar(45)
song_name varchar(45)
quantity
has
date_started
date_ended
subscribe to
upload
upload
has
Account_and_login_info
PK user_ID int
password varchar(45)
facebook_token varchar(45)
phone_number varchar(12)
is_active int
created_date date
last_active_date date
has
User_Info
PK user_ID int
first_name varchar(45)
last_name varchar(45)
weight_kg int
height_cm int
zodiac varchar(45)
birthdate date
education_level varchar(45)
drinking varchar(3)
smoking varchar(3)
kid_number int
religion varchar(45)
politics varchar(45)
instagram_token varchar(45)
spotify_token varchar(45)
user_action
date_seen
match with
participate
date
has
has
has
Official_Event
PK event_ID int
event_name varchar(45)
description varchar(1000)
date date
time time
online_or_in_person varchar(45)
Location
PK latitude_longitude varchar(45)
city varchar(45)
state_province varchar(45)
country varchar(45)
has
gender_orientation
PK idgender_orientation int
gender varchar(45)
sexual_orientation varchar(45)
gender_interested_in varchar(45)
has

Data Dictionary

Entity Name	Description	Aliases	Occurrence
Account_and_login_info	An entity that represents the login information and account status of a user's account.	Account	One account can link to one and only one user profile. One account can have one or many account subscriptions in its history (but only one subscription at a time).
User_info	An entity that represents the information that appears on users' profile to show his/her basic information and tastes.	User	One user can only have one and only one account. One user can post zero or many pictures, music, and hobbies. One user can have one or many pets (one pet type could be "none"). One user can have one and only one current location and gender orientation. One user can have zero or many locations if they are on the premium plan.
Subscription	An entity that represents the subscription provided by the application (free subscription or bumble boost/paid monthly plan)	N/A	One subscription can be chosen by one or many accounts.
Gender_orientation	An entity that represents the sexual orientation, gender, and interested gender of a user.	N/A	One gender and sexual orientation can be for one or many users. One gender orientation can be for zero or many events.
Pet	An entity that represents types of pets (e.g. dog) that may belong to users.	N/A	One type of pet belongs to one or many users ((one pet type could be "none").
Hobby	An entity that represents users' hobbies.	N/A	One hobby can be zero or many users' hobby.
Personal_pictures	An entity that represents pictures shared by users	N/A	One picture can be posted by one and only one user.
Music	An entity that represents music shared by users	N/A	One music can be shared by zero or many users.
Event	An entity that represents official bumble events that users can participate in.	N/A	One event can be for one or many gender orientations. One event can be participated by zero or many users. One event can have one and only one location.
Location	An entity that represents geological locations that can be users' locations and locations of events	N/A	One location can be shared by one or many users. One location be used by zero or many events and premium plan users.

Description of Attributes

Entity Name	**Attributes**	**Description**	**Data Type**	**Nulls**	**Multi-valued**	**Derived**	**Default**
Subscription	subscriptionID	Subscription unique identifier	INT	No	No	No	None
	subscription_name	Name of the subscription	VARCHAR(45)	No	No	No	None
	monthly_price	Price of the subscription	FLOAT	No	No	No	None
	max_swipes	Maximum number of swipes allowed	INT	No	No	No	None
	max_superswipes	Maximum number of super swipes allowed	INT	No	No	No	None
	multiple_locations	Can the user choose multiple locations for seeing people? (1 = yes, 0 = no)	INT	No	No	No	None
Account_has_subscription	user_ID	User's unique identifier	INT	No	No	No	None
	subscriptionID	Subscription unique identifier	INT	No	No	No	None
	date_started	Date the subscription started (unique identifier)	DATE	No	No	No	None
	date_ended	Date the subscription ended	DATE	Yes	No	No	None
Account_and_login_info	user_ID	User's unique identifier	INT	No	No	No	None
	password	User's password	VARCHAR(45)	No	No	No	None
	facebook_token	User's facebook profile link	VARCHAR(45)	Yes	No	No	None
	phone_number	User's phone number	VARCHAR(12)	No	No	No	None
	is_active	User's account is active or inactive (1 = yes, 0 = no)	INT	No	No	No	None
	created_date	User's account creation date	DATE	No	No	No	None

	last_active	User's account when last active	DATE	No	No	No	None
Pet	idPet	Pet type's unique identifier	INT	No	No	No	None
	pet_types	Type of pet	VARCHAR(45)	No	No	No	None
user_info_has_ pet	user_ID	User's unique identifier	INT	No	No	No	None
	idPet	Pet type's unique identifier	INT	No	Yes	No	None
	Quantity	Number of each type of pet a user has	INT	Yes	No	No	None
user_info	user_ID	User's unique identifier	INT	No	No	No	None
	first_name	User's first name	VARCHAR(45)	No	No	No	None
	last_name	User's last name	VARCHAR(45)	No	No	No	None
	weight_kg	User's weight in kg	INT	No	No	No	None
	height_cm	User's height in cm	INT	No	No	No	None
	zodiac	User's zodiac	VARCHAR(45)	No	No	No	None
	birthdate	User's birthdate	DATE	No	No	No	None
	education_level	User's education level	VARCHAR(45)	No	No	No	None
	drinking	Does the user drink	VARCHAR(3)	No	No	No	None
	smoking	Does the user smoke	VARCHAR(3)	No	No	No	None
	kid_number	Number of kids the user has	INT	No	No	No	None
	religion	User's religion	VARCHAR(45)	No	No	No	None
	politics	User's political view	VARCHAR(45)	No	No	No	None
	instagram_token	User's instagram account link	VARCHAR(45)	Yes	No	No	None
	spotify_token	User's spotify account link	VARCHAR(45)	Yes	No	No	None
	gender and orientation	User's gender and sexual orientation	INT	No	Yes	No	None
	current_latitude_ longitude	User's current location	INT	No	No	No	None

user_has_seen	user_ID1	User's unique identifier	INT	No	No	No	None
	user_ID2	User has seen another user's unique identifier	INT	No	No	No	None
	user_ID1_action	Action the user took on another user	VARCHAR(45)	No	No	No	None
	date_seen	Date a user has seen another user	DATE	No	No	No	None
user_info_has_ event	user_ID	User's unique identifier	INT	No	No	No	None
	event_id	Event's unique identifier	INT	No	Yes	No	None
personal_pictures	picture_url	Picture's unique identifier	VARCHAR(45)	No	No	No	None
	date_uploaded	Date the picture was uploaded	DATE	No	No	No	None
	size_KB	Picture's upload size in KB	INT	No	No	No	None
	user_ID	User's unique identifier	INT	No	No	No	None
user_has_music	user_ID	User's unique identifier	INT	No	No	No	None
	music_URL	Music's song unique identifier	VARCHAR(45)	No	Yes	No	None
music	music_url	Music's song unique identifier	VARCHAR(45)	No	No	No	None
	sung_by	Song is sung by	VARCHAR(45)	No	No	No	None
	time	Length of the song	TIME	No	No	No	None
	date_uploaded	Date the song was released	DATE	No	No	No	None
	genre_category	Music genre category	VARCHAR(45)	No	No	No	None
	album	Album the song is on	VARCHAR(45)	No	No	No	None
	song_name	Name of the song	VARCHAR(45)	No	No	No	None
Premium_plan_ locations	user_ID	User's unique identifier	INT	No	No	No	None

	latitude_longitude	User's unique location identifier	INT	No	Yes	No	None
	date	Date the user set the location (unique identifier)	DATE	No	No	No	None
location	latitude_longitude	Location's unique identifier	INT	No	No	No	None
	city	Location's city	VARCHAR(45)	No	No	No	None
	state_province	Location's state or province	VARCHAR(45)	No	No	No	None
	country	Location's country	VARCHAR(45)	No	No	No	None
gender_orientation	idgender_orientation	Gender and orientation unique identifier	INT	No	No	No	None
	sexual_orientation	User's sexual orientation	VARCHAR(45)	No	No	No	None
	gender	User's gender	VARCHAR(45)	No	No	No	None
	gender_interested_in	User is interested in which gender	VARCHAR(45)	No	No	No	None
event_has_gender_orientation	event_id	Event's unique identifier	INT	No	No	No	None
	idgender_orientation	Event is for specific gender and orientation	INT	No	Yes	No	None
event	event_id	Event's unique identifier	INT	No	No	No	None
	event_name	Name of the event	VARCHAR(45)	No	No	No	None
	description	Description of the event	VARCHAR(1000)	No	No	No	None
	date	Date of the event	DATE	No	No	No	None
	time	Time the event starts	TIME	No	No	No	None
	online_or_in_person	Is the event online or in person	VARCHAR(45)	No	No	No	None
	latitude_longitude	Location of the event	VARCHAR(45)	No	No	No	None

Relational Schema

Logical Model

Account_and_login_info (user_ID, password, facebook_token, phone_number, is_active, created_date, last_active_date)
PK: user_ID
FK: N/A

Location (latitude_longitude, city, state_province, country)
PK: latitude_longitude
FK: N/A

Account_has_subscription (user_ID, subscriptionID, date_started, date_ended)
PK: user_ID, subscriptionID
FK: user_ID references Account_and_login_info (user_ID)
subscriptionID reference Subscription (subscriptionID)

Subscription (subscriptionID, subscription_name, monthly_price, max_swipes_daily, max_superswipes_weekly, multiple_location)
PK: subscriptionID
FK: N/A

Gender_orientation (idgender_orientation, gender, sexual_orientation, gender_interested_in)
PK: idgender_orientation
FK: N/A

User_info (user_ID, first_name, last_name, weight_kg, height_cm, zodiac, birthdate, education_level, drinking, smoking, kid_number, religion, politics, instagram_token, spotify_token, idgender_orientation, current_latitude_longitude)
PK: user_ID
FK: user_ID references Account_and_login_info (user_ID)
idgender_orientation references Gender_orientation (idgender_orientation)
current_latitude_longitude references Location (latitude_longitude)

User_info_has_event (user_ID, event_id)
PK: user_ID, event_id
FK: user_ID references User_info (user_ID)
event_id references Event (event_id)

Event (event_id, event_name, description, date, time, online_or_in_person, latitude_longitude)
PK: event_id
FK: latitude_longitude references Location (latitude_longitude)

Event_has_gender_orientation (event_id, idgender_orientation)
PK: event_id, idgender_orientation
FK: event_id references Event (event_id)
idgender_orientation references Gender_orientation (idgender_orientation)

Premium_Plan_Locations (user_ID, latitude_longitude, date)
PK: user_ID, latitude_longitude, date
FK: user_ID references User_info (user_ID)
latitude_longitude references Location (latitude_longitude)

Music (music_URL, sung_by, time, date_uploaded, genre_category, album, song_name)
PK: music_URL
FK: N/A

User_has_music (user_ID, music_URL)
PK: user_ID, music_URL
FK: user_ID references User_info (user_ID)
music_URL references Music (music_URL)

Personal_pictures (picture_URL, date_uploaded, size, user_ID)
PK: picture_URL, user_ID
FK: user_ID references User_info (user_ID)

Pet (idPet, pet_type)
PK: idPet
FK: N/A

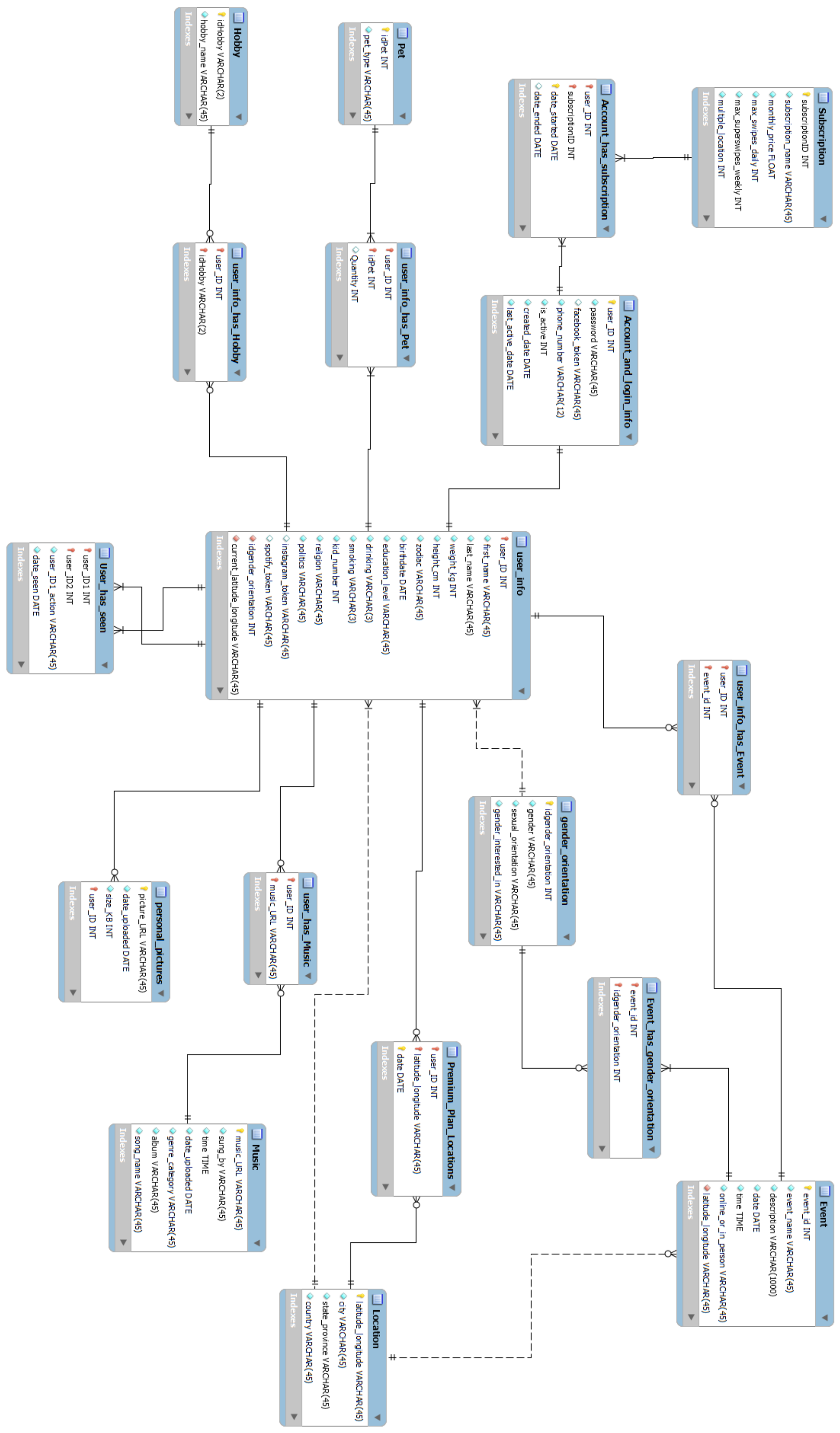
User_info_has_pet (user_ID, idPet)
PK: user_ID, idPet
FK: user_ID references User_info (user_ID)
idPet references Pet(idPet)

Hobby (idHobby, hobby_name)
PK: idHobby
FK: N/A

User_info_has_Hobby (user_ID, idHobby)
PK: user_ID, idHobby
FK: user_ID references User_info (user_ID)
idHobby references Hobby (idHobby)

User_has_seen (user_ID1, user_ID2, user_ID1_action, date_seen)
PK: user_ID1, user_ID2
FK: user_ID1 references User_info (user_ID)
user_ID2 references User_info (user_ID)

Physical Model



Section 2 – Queries

Theme 1- App features

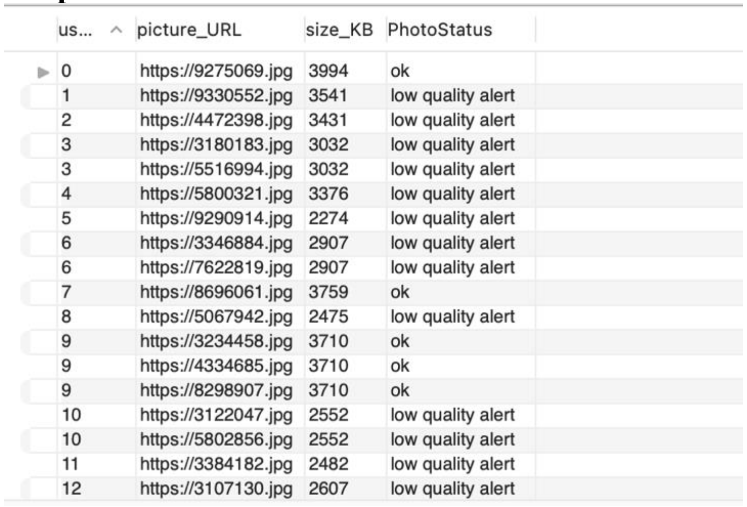
Query 1

Objective: To highlight pictures with the wrong format (low quality or heavy file) with the idea of displaying an alert for low quality images and a process to compress heavy files.

Assumptions: HD photos (1270 x 720 pixels) in 32-bit (CMYK 16.7 million colors) format are about 3600 Kb, while Ultra HD photos (1920 x 1080 pixels) in the same format are 8,294 Kb, therefore we assume any pictures between 3600 kb and 5700 kb to be the correct format and under that bound as “low quality” and above as “to be compressed”.

Code:
```
/* Photo Quality alert*/
select pp.user_ID,  pp.picture_URL, pp.size_KB,
case
        when pp.size_KB between 3600 and 5700 then "ok"
        when pp.size_KB > 5700 then "compression required"
        else "low quality alert"
end as PhotoStatus
from personal_pictures as pp
order by user_id;
```

Output:



us...	picture_URL	size_KB	PhotoStatus
0	https://9275069.jpg	3994	ok
1	https://9330552.jpg	3541	low quality alert
2	https://4472398.jpg	3431	low quality alert
3	https://3180183.jpg	3032	low quality alert
3	https://5516994.jpg	3032	low quality alert
4	https://5800321.jpg	3376	low quality alert
5	https://9290914.jpg	2274	low quality alert
6	https://3346884.jpg	2907	low quality alert
6	https://7622819.jpg	2907	low quality alert
7	https://8696061.jpg	3759	ok
8	https://5067942.jpg	2475	low quality alert
9	https://3234458.jpg	3710	ok
9	https://4334685.jpg	3710	ok
9	https://8298907.jpg	3710	ok
10	https://3122047.jpg	2552	low quality alert
10	https://5802856.jpg	2552	low quality alert
11	https://3384182.jpg	2482	low quality alert
12	https://3107130.jpg	2607	low quality alert

Interpretation of results: This query rapidly categorises which picture are low quality or to be compressed (too heavy). In this manner, we will be able to rapidly flag the “wrong” pictures and invite the user to change them. Moreover, for the heavy pictures, we could image a compression process that would automatically crunch the heavy pictures into lower quality versions.

Theme 2 – Feature exploration

Query 2

Objective: To see which gender and sexual orientation group has been the most successful in matches.

Assumptions: A successful match is when two people both right swipe on each other, either one of them swipes right and the other super swipes, or both super swipe each other. The percentage of matches is calculated by the total matches of a particular gender and orientation group divided by the total swipes done by that gender and sexual orientation group.

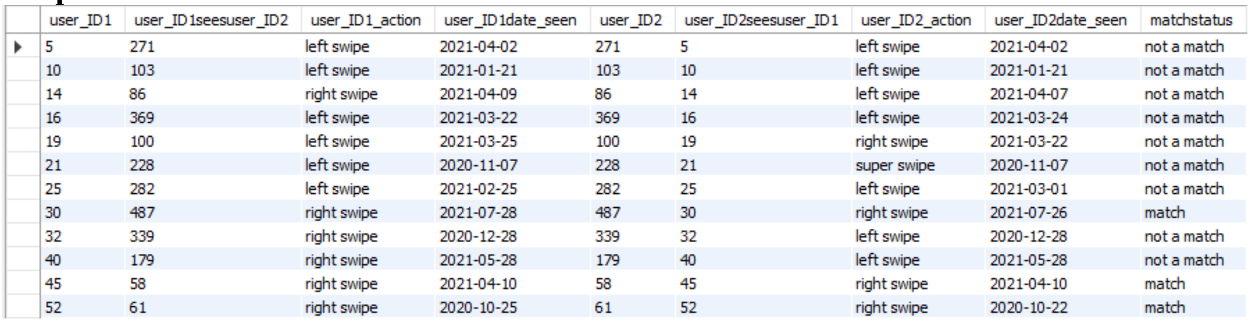
Code:

```
create view matchstatus as
select A.user_ID1,  A.user_ID2 as user_ID1seesuser_ID2, A.user_ID1_action, A.date_seen as
user_ID1date_seen, B.user_ID1 as user_ID2,  B.user_ID2 as user_ID2seesuser_ID1,
B.user_ID1_action as user_ID2_action,B.date_seen as user_ID2date_seen,
case
        when A.user_ID1_action="right swipe" and (B.user_ID1_action="right swipe" or
        B.user_ID1_action="super swipe") then "match"
        when A.user_ID1_action="super swipe" and (B.user_ID1_action="right swipe" or
        B.user_ID1_action="super swipe") then "match"
        when B.user_ID1_action="right swipe" and (A.user_ID1_action="right swipe" or
        A.user_ID1_action="super swipe") then "match"
        when B.user_ID1_action="super swipe" and (A.user_ID1_action="right swipe" or
        A.user_ID1_action="super swipe") then "match"
else "not a match"
end as matchstatus
from user_has_seen as A, user_has_seen as B
where A.user_ID1=B.user_ID2 and A.user_ID2=B.user_ID1 ;

/* straight women*/
select "straight women" as genderorientation, concat(round(count(*)/totalidegender*100,2),"%")
as "Percentage of matches in that gender/orientation group"
from matchstatus, user_info, gender_orientation, (select count(*) as totalidegender from
matchstatus, user_info where user_info.idgender_orientation=2001 and
matchstatus.user_ID1=user_info.user_ID) as nummatches
where matchstatus.user_ID1=user_info.user_ID and
user_info.idgender_orientation=gender_orientation.idgender_orientation and
user_info.idgender_orientation=2001 and matchstatus="match"

union

/* straight men*/
select "straight men" as genderorientation, concat(round(count(*)/totalidegender*100,2),"%") as
"Percentage of matches in that gender/orientation group"
```

```
from matchstatus, user_info, gender_orientation, (select count(*) as totalidegender from
matchstatus, user_info where user_info.idgender_orientation=2002 and
matchstatus.user_ID1=user_info.user_ID) as nummatches
where matchstatus.user_ID1=user_info.user_ID and
user_info.idgender_orientation=gender_orientation.idgender_orientation and
user_info.idgender_orientation=2002 and matchstatus="match"

union

/*gay men*/
select "gay men" as genderorientation, concat(round(count(*)/totalidegender*100,2),"%") as
"Percentage of matches in that gender/orientation group"
from matchstatus, user_info, gender_orientation, (select count(*) as totalidegender from
matchstatus, user_info where user_info.idgender_orientation=2004 and
matchstatus.user_ID1=user_info.user_ID) as nummatches
where matchstatus.user_ID1=user_info.user_ID and
user_info.idgender_orientation=gender_orientation.idgender_orientation and
user_info.idgender_orientation=2004 and matchstatus="match"

union

/* lesbian women*/
select "lesbian women" as genderorientation, concat(round(count(*)/totalidegender*100,2),"%")
as "Percentage of matches in that gender/orientation group"
from matchstatus, user_info, gender_orientation, (select count(*) as totalidegender from
matchstatus, user_info where user_info.idgender_orientation=2006 and
matchstatus.user_ID1=user_info.user_ID) as nummatches
where matchstatus.user_ID1=user_info.user_ID and
user_info.idgender_orientation=gender_orientation.idgender_orientation and
user_info.idgender_orientation=2006 and matchstatus="match"

union

/* bi women*/
select "bi-sexual women" as genderorientation,
concat(round(count(*)/totalidegender*100,2),"%") as "Percentage of matches in that
gender/orientation group"
from matchstatus, user_info, gender_orientation, (select count(*) as totalidegender from
matchstatus, user_info where user_info.idgender_orientation=2003 and
matchstatus.user_ID1=user_info.user_ID) as nummatches
where matchstatus.user_ID1=user_info.user_ID and
user_info.idgender_orientation=gender_orientation.idgender_orientation and
user_info.idgender_orientation=2003 and matchstatus="match"

union
/* bi men*/
```

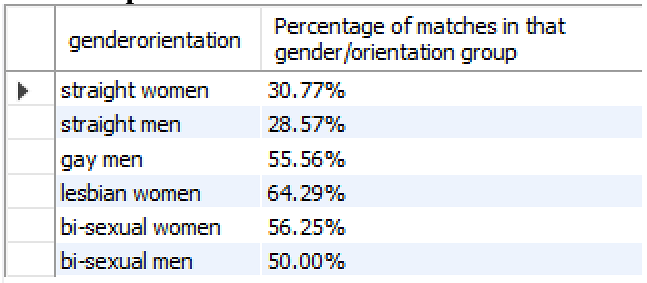
```
select "bi-sexual men" as genderorientation, concat(round(count(*)/totalidegender*100,2),"%")
as "Percentage of matches in that gender/orientation group"
from matchstatus, user_info, gender_orientation, (select count(*) as totalidegender from
matchstatus, user_info where user_info.idgender_orientation=2005 and
matchstatus.user_ID1=user_info.user_ID) as nummatches
where matchstatus.user_ID1=user_info.user_ID and
user_info.idgender_orientation=gender_orientation.idgender_orientation and
user_info.idgender_orientation=2005 and matchstatus="match"
```

Output from the view table "matchstatus":



user_ID1	user_ID1seesuser_ID2	user_ID1_action	user_ID1date_seen	user_ID2	user_ID2seesuser_ID1	user_ID2_action	user_ID2date_seen	matchstatus
5	271	left swipe	2021-04-02	271	5	left swipe	2021-04-02	not a match
10	103	left swipe	2021-01-21	103	10	left swipe	2021-01-21	not a match
14	86	right swipe	2021-04-09	86	14	left swipe	2021-04-07	not a match
16	369	left swipe	2021-03-22	369	16	left swipe	2021-03-24	not a match
19	100	left swipe	2021-03-25	100	19	right swipe	2021-03-22	not a match
21	228	left swipe	2020-11-07	228	21	super swipe	2020-11-07	not a match
25	282	left swipe	2021-02-25	282	25	left swipe	2021-03-01	not a match
30	487	right swipe	2021-07-28	487	30	right swipe	2021-07-26	match
32	339	right swipe	2020-12-28	339	32	left swipe	2020-12-28	not a match
40	179	right swipe	2021-05-28	179	40	left swipe	2021-05-28	not a match
45	58	right swipe	2021-04-10	58	45	right swipe	2021-04-10	match
52	61	right swipe	2020-10-25	61	52	right swipe	2020-10-22	match

Final output:



genderorientation	Percentage of matches in that gender/orientation group
straight women	30.77%
straight men	28.57%
gay men	55.56%
lesbian women	64.29%
bi-sexual women	56.25%
bi-sexual men	50.00%

Interpretation of results: Approximately 64% of lesbian women have successfully had at least one match on Bumble, making them the most likely gender and orientation group to get a match. Approximately 28% of straight men have successfully had at least one match on Bumble, making them the least likely gender and orientation group to get a match.

Query 3

Objective: Find out the distribution of registered user's zodiac sign.

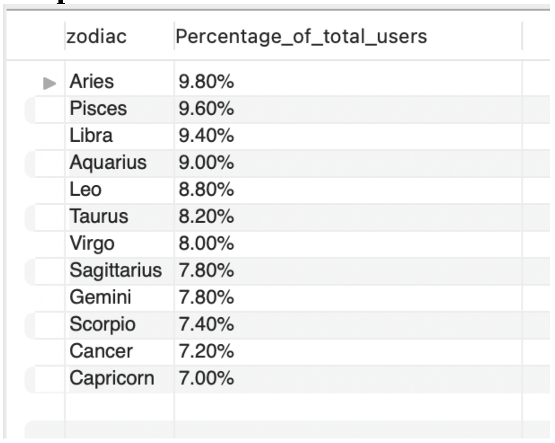
Assumptions: A user can only have one zodiac sign.

Code:

```
SELECT zodiac, concat(TRUNCATE((COUNT(*)*100/ (
        SELECT COUNT(*) AS total_users
        FROM user_info
        )),2),"%") AS Percentage_of_total_users
```

```
FROM user_info
GROUP BY zodiac
ORDER BY Percentage_of_total_users desc;
```

Output:



zodiac	Percentage_of_total_users
Aries	9.80%
Pisces	9.60%
Libra	9.40%
Aquarius	9.00%
Leo	8.80%
Taurus	8.20%
Virgo	8.00%
Sagittarius	7.80%
Gemini	7.80%
Scorpio	7.40%
Cancer	7.20%
Capricorn	7.00%

Interpretation of results: There is an almost homogenous mixture of all the zodiac signs with 9.8% of users having the Aries zodiac sign. Users with belonging to the Capricorn zodiac are least likely on the app forming only 7% of the total registered users.

Query 4

Objective: Find out the time of the year - month and week when highest number of matches occur.

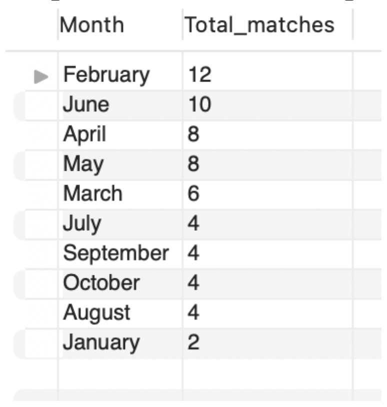
Assumptions: User A & User B are considered a match when both right swipe on each other, either one of them swipes right and the other super swipes, or both super swipe each other. The matched date is the maximum of when User A saw User B and vice-versa. Total number of matches occurring in a month or day of the week is taken a measure to find the best time to get a match. This query uses the "matches" view table from query 8. ***Please run query 8 first before running query 4***.

Code:

```
SELECT
CASE
        WHEN MONTH(matchdate) = 1 THEN "January"
        WHEN MONTH(matchdate) = 2 THEN "February"
        WHEN MONTH(matchdate) = 3 THEN "March"
        WHEN MONTH(matchdate) = 4 THEN "April"
        WHEN MONTH(matchdate) = 5 THEN "May"
        WHEN MONTH(matchdate) = 6 THEN "June"
        WHEN MONTH(matchdate) = 7 THEN "July"
        WHEN MONTH(matchdate) = 8 THEN "August"
        WHEN MONTH(matchdate) = 9 THEN "September"
```

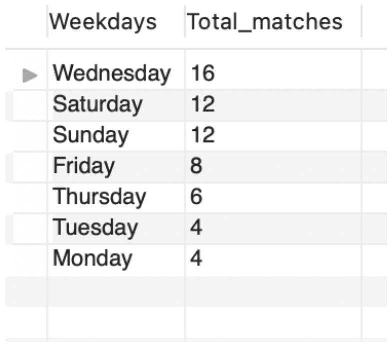
```
            WHEN MONTH(matchdate) = 10 THEN "October"
            WHEN MONTH(matchdate) = 11 THEN "November"
            ELSE "December"
END AS Month, COUNT(*) AS Total_matches
from
(SELECT *,
CASE
WHEN user_ID1date_seen> user_ID2date_seen THEN user_ID1date_seen
    ELSE user_ID2date_seen
END AS matchdate
FROM matches) as matchesupdated
GROUP BY month
ORDER BY Total_matches DESC;

/* by weekday*/
SELECT
CASE
            WHEN weekday(matchdate) = 0 THEN "Monday"
            WHEN weekday(matchdate) = 1 THEN "Tuesday"
            WHEN weekday(matchdate) = 2 THEN "Wednesday"
            WHEN weekday(matchdate) = 3 THEN "Thursday"
            WHEN weekday(matchdate) = 4 THEN "Friday"
            WHEN weekday(matchdate) = 5 THEN "Saturday"
            WHEN weekday(matchdate) = 6 THEN "Sunday"
            ELSE NULL
END AS Weekdays, COUNT(*) AS Total_matches
FROM (SELECT *,
CASE
            WHEN user_ID1date_seen> user_ID2date_seen THEN user_ID1date_seen
            ELSE user_ID2date_seen
END AS matchdate
FROM matches) as matchesupdated
GROUP BY Weekdays
ORDER BY Total_matches DESC ;
```

Output of the month comparison



Month	Total_matches
February	12
June	10
April	8
May	8
March	6
July	4
September	4
October	4
August	4
January	2

Output of the day of the week comparison



Weekdays	Total_matches
Wednesday	16
Saturday	12
Sunday	12
Friday	8
Thursday	6
Tuesday	4
Monday	4

Interpretation of results: From the query result, we can observe that the highest number of matches happen in February and on Wednesday whereas a user is least likely to get a match on Mondays and in January.

Query 5

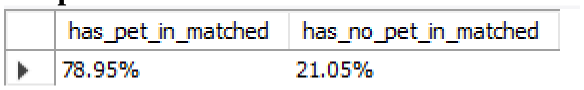
Objective: This query is to check if people with pets match more than people who have no pet (are people who have pets more popular on our App?).

Assumptions: This query is based on people who made successful matches instead of on matches.

Code:
```
SELECT concat(round(pet.pet_count/all_included.all_count*100,2),"%") AS has_pet_in_matched,
concat(round((1-pet.pet_count/all_included.all_count)*100,2),"%") AS has_no_pet_in_matched
FROM(
        SELECT COUNT(pett.has_pet) AS pet_count
        FROM(
                SELECT i.user_ID AS has_pet
                FROM user_info AS i
                INNER JOIN user_info_has_pet AS ipet ON i.user_ID = ipet.user_ID
                WHERE ipet.quantity IS NOT NULL
                AND i.user_ID IN (
                SELECT DISTINCT user_ID1
                FROM(
                        SELECT A.user_ID1
                        FROM user_has_seen AS A, user_has_seen AS B
                        WHERE A.user_ID1=B.user_ID2
                        AND A.user_ID2=B.user_ID1
                        AND ((A.user_ID1_action="right swipe" AND (B.user_ID1_action="right
                        swipe" OR B.user_ID1_action="super swipe"))
                                OR
                        (A.user_ID1_action="super swipe" AND (B.user_ID1_action="right swipe" OR
                        B.user_ID1_action="super swipe")))
                        UNION
                        SELECT A.user_ID2
```

```
            FROM user_has_seen AS A, user_has_seen AS B
            WHERE A.user_ID1=B.user_ID2
            AND A.user_ID2=B.user_ID1
            AND ((A.user_ID1_action="right swipe" AND (B.user_ID1_action="right
            swipe" OR B.user_ID1_action="super swipe"))
                OR
            (A.user_ID1_action="super swipe" AND (B.user_ID1_action="right swipe" OR
            B.user_ID1_action="super swipe")))
            ) AS unioned
        )
        GROUP BY i.user_ID) AS pett
    ) AS pet
,
    (SELECT COUNT(alll.all_matched_count) AS all_count
    FROM(
        SELECT i.user_ID AS all_matched_count
        FROM user_info AS i
        INNER JOIN user_info_has_pet AS ipet ON i.user_ID = ipet.user_ID
        WHERE i.user_ID IN (
        SELECT DISTINCT user_ID1
        FROM(
            SELECT A.user_ID1
            FROM user_has_seen AS A, user_has_seen AS B
            WHERE A.user_ID1=B.user_ID2
            AND A.user_ID2=B.user_ID1
            AND ((A.user_ID1_action="right swipe" AND (B.user_ID1_action="right
            swipe" OR B.user_ID1_action="super swipe"))
                OR
            (A.user_ID1_action="super swipe" AND (B.user_ID1_action="right swipe" OR
            B.user_ID1_action="super swipe")))
            UNION
            SELECT A.user_ID2
            FROM user_has_seen AS A, user_has_seen AS B
            WHERE A.user_ID1=B.user_ID2
            AND A.user_ID2=B.user_ID1
            AND ((A.user_ID1_action="right swipe" AND (B.user_ID1_action="right
            swipe" OR B.user_ID1_action="super swipe"))
                OR
            (A.user_ID1_action="super swipe" AND (B.user_ID1_action="right swipe" OR
            B.user_ID1_action="super swipe")))
            ) AS unioned
        )
        GROUP BY i.user_ID) AS alll
    ) AS all_included;
```

Output:



	has_pet_in_matched	has_no_pet_in_matched
▶	78.95%	21.05%

Interpretation of results: 78.95% of the people who made successful matches had pets while 21.05% of them did not, meaning that people with pets were more popular.

Query 6

Objective: To find which height pairings of users is most frequently seen among the successful matches.

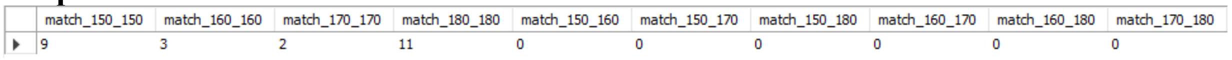
Assumptions: We divided the heights of users into 4 categories:

1. 150cm to 159 cm (noted as “150”)
2. 160cm to 169 cm (noted as “160”)
3. 170cm to 179 cm (noted as “170”)
4. 180cm to 190 cm (noted as “180”)

Code:

```
/* create view tables to store generated value for later query */
CREATE VIEW height_match AS(
        SELECT u.user_ID AS ID1, u.height_cm AS h1, B2.ID2 AS ID2, B2.h2 AS h2
        FROM user_info as u
        INNER JOIN (
                SELECT A.user_ID1 AS user1
                FROM user_has_seen AS A, user_has_seen AS B
                WHERE A.user_ID1=B.user_ID2
                AND A.user_ID2=B.user_ID1
                AND ((A.user_ID1_action="right swipe" AND (B.user_ID1_action="right swipe" OR
                B.user_ID1_action="super swipe"))
                OR
                (A.user_ID1_action="super swipe" AND (B.user_ID1_action="right swipe" OR
                B.user_ID1_action="super swipe")))
                ) AS A ON A.user1 = u.user_ID
        INNER JOIN (
                SELECT u.user_ID AS ID2, u.height_cm AS h2, B.u1 AS user_match
                FROM user_info as u
                INNER JOIN (
                        SELECT A.user_ID2 AS user2, A.user_ID1 AS u1
                        FROM user_has_seen AS A, user_has_seen AS B
                        WHERE A.user_ID1=B.user_ID2
                        AND A.user_ID2=B.user_ID1
                        AND ((A.user_ID1_action="right swipe" AND (B.user_ID1_action="right
                        swipe" OR B.user_ID1_action="super swipe"))
                                OR
                        (A.user_ID1_action="super swipe" AND (B.user_ID1_action="right swipe" OR
                        B.user_ID1_action="super swipe")))
                        ) AS B ON B.user2 = u.user_ID
        ) AS B2 ON B2.user_match = A.user1
);

CREATE VIEW height_matched AS (
SELECT
        (CASE WHEN ((h1>=150 AND h1<=159) AND (h2>=150 AND h2<=159)) THEN 1 ELSE 0
        END) AS match_150_150,
        (CASE WHEN ((h1>=160 AND h1<=169) AND (h2>=160 AND h2<=169)) THEN 1 ELSE 0
        END) AS match_160_160,
        (CASE WHEN ((h1>=170 AND h1<=179) AND (h2>=170 AND h2<=179)) THEN 1 ELSE 0
        END) AS match_170_170,
        (CASE WHEN ((h1>=180 AND h1<=190) AND (h2>=180 AND h2<=190)) THEN 1 ELSE 0
        END) AS match_180_180,
        (CASE WHEN (((h1>=150 AND h1<=159) AND (h2>=160 AND h2<=169)) AND ((h1>=160
        AND h1<=169) AND (h2>=150 AND h2<=159))) THEN 1 ELSE 0 END) AS match_150_160,
        (CASE WHEN (((h1>=150 AND h1<=159) AND (h2>=170 AND h2<=179)) AND ((h1>=170
        AND h1<=179) AND (h2>=150 AND h2<=159))) THEN 1 ELSE 0 END) AS match_150_170,
        (CASE WHEN (((h1>=150 AND h1<=159) AND (h2>=180 AND h2<=190)) AND ((h1>=180
        AND h1<=190) AND (h2>=150 AND h2<=159))) THEN 1 ELSE 0 END) AS match_150_180,
        (CASE WHEN (((h1>=160 AND h1<=169) AND (h2>=170 AND h2<=179)) AND ((h1>=170
        AND h1<=179) AND (h2>=160 AND h2<=169))) THEN 1 ELSE 0 END) AS match_160_170,
        (CASE WHEN (((h1>=160 AND h1<=169) AND (h2>=180 AND h2<=190)) AND ((h1>=180
        AND h1<=190) AND (h2>=160 AND h2<=169))) THEN 1 ELSE 0 END) AS match_160_180,
        (CASE WHEN (((h1>=170 AND h1<=179) AND (h2>=180 AND h2<=190)) AND ((h1>=180
        AND h1<=190) AND (h2>=170 AND h2<=179))) THEN 1 ELSE 0 END) AS match_170_180
FROM height_match
);

/* query to compare pairs of heights in matches */
SELECT
        SUM(match_150_150) AS match_150_150, SUM(match_160_160) AS match_160_160,
        SUM(match_170_170) AS match_170_170, SUM(match_180_180) AS match_180_180,
          SUM(match_150_160) AS match_150_160, SUM(match_150_170) AS match_150_170,
        SUM(match_150_180) AS match_150_180, SUM(match_160_170) AS match_160_170,
          SUM(match_160_180) AS match_160_180, SUM(match_170_180) AS match_170_180
FROM height_matched;
```

Output:



match_150_150	match_160_160	match_170_170	match_180_180	match_150_160	match_150_170	match_150_180	match_160_170	match_160_180	match_170_180
9	3	2	11	0	0	0	0	0	0

Interpretation of results: According to the result table, matches between users who both have height in the range of 180 cm to 190 cm are most frequently seen in all the matches. Matches between two people who have both 150-159 cm height are the second frequent.

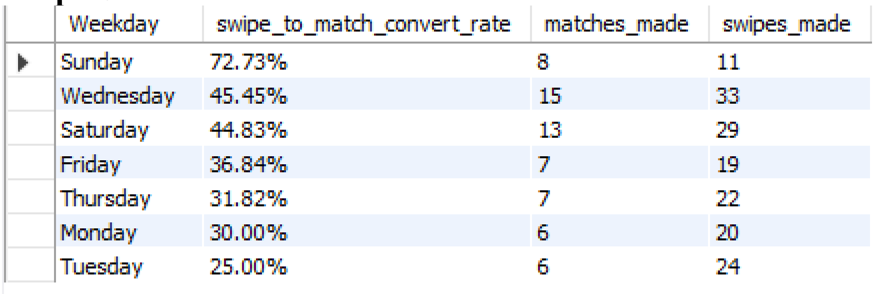
Query 7

Objective: This query aims to compare the swipe-to-matches convert rate of swipes that happened on each weekday, i.e., if a person made a swipe on a certain day of week, what is the possibility of that swipe to end up in a successful match.

Code:

```
SELECT
(CASE
    WHEN matched.weekdays = 0 THEN "Monday"
              WHEN matched.weekdays = 1 THEN "Tuesday"
              WHEN matched.weekdays = 2 THEN "Wednesday"
              WHEN matched.weekdays = 3 THEN "Thursday"
              WHEN matched.weekdays = 4 THEN "Friday"
              WHEN matched.weekdays = 5 THEN "Saturday"
              WHEN matched.weekdays = 6 THEN "Sunday"
              ELSE NULL
              END) AS Weekday,
CONCAT(ROUND(matched.match_date/swipe_date*100,2),"%") AS swipe_to_match_convert_rate,
SUM(matched.match_date) AS matches_made, SUM(swipes.swipe_date) AS swipes_made
FROM (
          SELECT WEEKDAY(A.date_seen) AS weekdays, COUNT(WEEKDAY(A.date_seen)) AS
          match_date
          FROM user_has_seen AS A, user_has_seen AS B
          WHERE A.user_ID1=B.user_ID2
          AND A.user_ID2=B.user_ID1
          AND ((A.user_ID1_action="right swipe" AND (B.user_ID1_action="right swipe" OR
          B.user_ID1_action="super swipe"))
                    OR
          (A.user_ID1_action="super swipe" AND (B.user_ID1_action="right swipe" OR
          B.user_ID1_action="super swipe")))
              GROUP BY WEEKDAY(A.date_seen)
              ) AS matched
INNER JOIN (
          SELECT WEEKDAY(date_seen) AS weekdays, COUNT(WEEKDAY(date_seen)) AS
          swipe_date
          FROM user_has_seen
          GROUP BY WEEKDAY(date_seen)
          ) AS swipes ON matched.weekdays = swipes.weekdays
GROUP BY matched.weekdays
ORDER BY matched.match_date/swipe_date DESC;
```

Output:



Weekday	swipe_to_match_convert_rate	matches_made	swipes_made
Sunday	72.73%	8	11
Wednesday	45.45%	15	33
Saturday	44.83%	13	29
Friday	36.84%	7	19
Thursday	31.82%	7	22
Monday	30.00%	6	20
Tuesday	25.00%	6	24

Interpretation of results: Based on our historical data (i.e., successful previous matches), if one person made a swipe on Sunday, there is a 0.7273 of chance of that swipe to lead to a successful match. Sunday has the highest conversion rate while Tuesday has the lowest.

Theme 3 – Matching / Combination Metrics

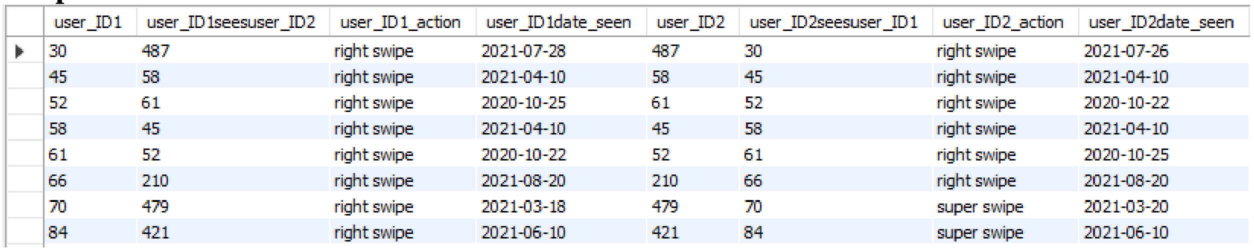
Query 8

Objective: Get a list of all the people that right swiped / super swiped on each other. This will help us see if the app is working and that matches do happen.

Assumptions: A successful match is when two people both right swipe on each other, either one of them swipes right and the other super swipes, or both super swipe each other.

Code:
```
create view matches as
select A.user_ID1,  A.user_ID2 as user_ID1seesuser_ID2, A.user_ID1_action, A.date_seen as user_ID1date_seen, B.user_ID1 as user_ID2,  B.user_ID2 as user_ID2seesuser_ID1, B.user_ID1_action as user_ID2_action, B.date_seen as user_ID2date_seen
from user_has_seen as A, user_has_seen as B
where A.user_ID1=B.user_ID2 and A.user_ID2=B.user_ID1 and (A.user_ID1_action="right swipe" or A.user_ID1_action="super swipe") and (B.user_ID1_action="right swipe" or B.user_ID1_action="super swipe");
```

Output:



user_ID1	user_ID1seesuser_ID2	user_ID1_action	user_ID1date_seen	user_ID2	user_ID2seesuser_ID1	user_ID2_action	user_ID2date_seen
30	487	right swipe	2021-07-28	487	30	right swipe	2021-07-26
45	58	right swipe	2021-04-10	58	45	right swipe	2021-04-10
52	61	right swipe	2020-10-25	61	52	right swipe	2020-10-22
58	45	right swipe	2021-04-10	45	58	right swipe	2021-04-10
61	52	right swipe	2020-10-22	52	61	right swipe	2020-10-25
66	210	right swipe	2021-08-20	210	66	right swipe	2021-08-20
70	479	right swipe	2021-03-18	479	70	super swipe	2021-03-20
84	421	right swipe	2021-06-10	421	84	super swipe	2021-06-10

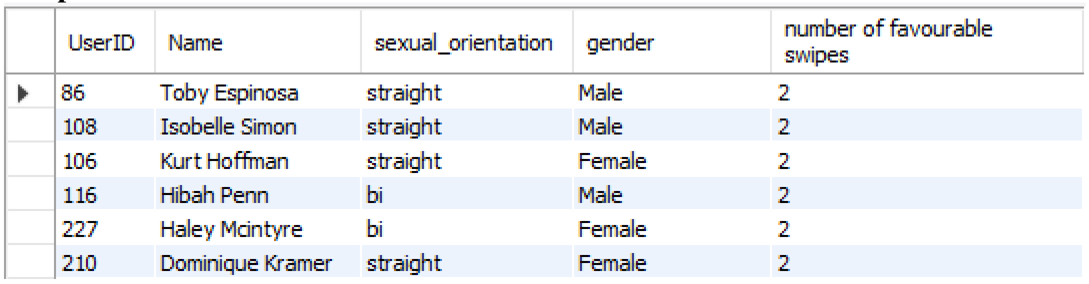
Query 9

Objective: To see which users were swiped right / super swiped on the most.

Assumptions: A successful match is when two people both right swipe on each other, either one of them swipes right and the other super swipes, or both super swipe each other.

Code:
```
select user_ID2 as UserID, concat(user_info.first_name," ", user_info.last_name) as Name, sexual_orientation, gender, count(*) as "number of favourable swipes"
from user_info, user_has_seen, gender_orientation
where (user_has_seen.user_ID1_action="right swipe" or user_has_seen.user_ID1_action="super_swipe") and user_info.user_ID=user_has_seen.user_ID2 and gender_orientation.idgender_orientation=user_info.idgender_orientation
group by user_ID2
order by count(*) desc limit 6;
```

Output:



UserID	Name	sexual_orientation	gender	number of favourable swipes
86	Toby Espinosa	straight	Male	2
108	Isobelle Simon	straight	Male	2
106	Kurt Hoffman	straight	Female	2
116	Hibah Penn	bi	Male	2
227	Haley Mcintyre	bi	Female	2
210	Dominique Kramer	straight	Female	2

Interpretation of results: There are six top users with the most right / super swipes, with a total of two favourable swipes, most of them being straight. In reality, a user can be swiped right / super swiped on much more than two times. However, due to the scope of this project being to demonstrate the feasibility of the database design, we did not incorporate a lot of data.

Theme 4 – Performance Metrics

Query 10

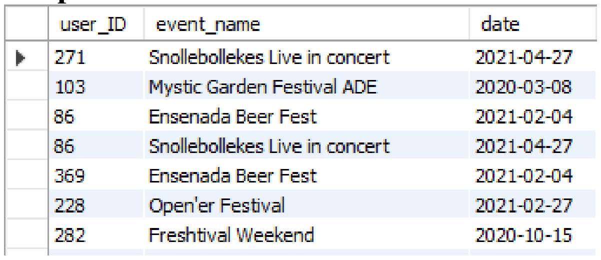
Objective: To see all the people that attended an event and if there were people that matched at some point.

Assumptions: The query uses the "matches" view table from query 8.

Code:
```
/* Get the list of events for each userID in user has seen*/
create view userevent as
select user_ID1 as user_ID, event_name, date
from user_has_seen, user_info, user_info_has_event, event
where user_has_seen.user_ID1=user_info.user_ID and
user_info.user_ID=user_info_has_event.user_ID and
user_info_has_event.event_id=event.event_id;

/* Display the events and matches*/
select userid1event.user_ID1+userid1event.user_ID2 as matchID, userid1event.user_ID1,
userid1event.user_ID1date_seen, userid1event.user_ID1event, userid1event.user_ID1eventdate,
userif2event.user_ID2, userif2event.user_ID2date_seen, userif2event.user_ID2event,
userif2event.user_ID2eventdate
from
/* userid1 events*/
(select user_ID1, user_ID2, user_ID1date_seen, event_name as user_ID1event, date as
user_ID1eventdate
from matches, userevent
where user_ID1=user_ID) as userid1event,
/*userid2 events*/
```

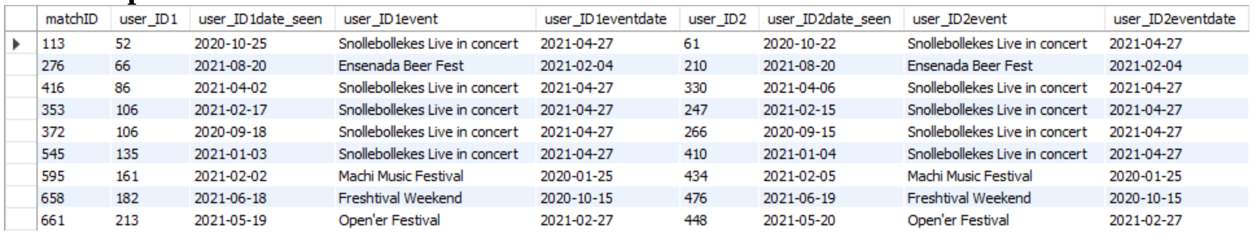
```
(select user_ID1, user_ID2, user_ID2date_seen, event_name as user_ID2event, date as
user_ID2eventdate
from matches, userevent
where user_ID2=user_ID) as userif2event
where userid1event.user_ID2=userif2event.user_ID2 and
userid1event.user_ID1=userif2event.user_ID1 and
userid1event.user_ID1event=userif2event.user_ID2event
group by matchID;
```

Output from the view table "userevent":



user_ID	event_name	date
271	Snollebollekes Live in concert	2021-04-27
103	Mystic Garden Festival ADE	2020-03-08
86	Ensenada Beer Fest	2021-02-04
86	Snollebollekes Live in concert	2021-04-27
369	Ensenada Beer Fest	2021-02-04
228	Open'er Festival	2021-02-27
282	Freshtival Weekend	2020-10-15

Final output:



matchID	user_ID1	user_ID1date_seen	user_ID1event	user_ID1eventdate	user_ID2	user_ID2date_seen	user_ID2event	user_ID2eventdate
113	52	2020-10-25	Snollebollekes Live in concert	2021-04-27	61	2020-10-22	Snollebollekes Live in concert	2021-04-27
276	66	2021-08-20	Ensenada Beer Fest	2021-02-04	210	2021-08-20	Ensenada Beer Fest	2021-02-04
416	86	2021-04-02	Snollebollekes Live in concert	2021-04-27	330	2021-04-06	Snollebollekes Live in concert	2021-04-27
353	106	2021-02-17	Snollebollekes Live in concert	2021-04-27	247	2021-02-15	Snollebollekes Live in concert	2021-04-27
372	106	2020-09-18	Snollebollekes Live in concert	2021-04-27	266	2020-09-15	Snollebollekes Live in concert	2021-04-27
545	135	2021-01-03	Snollebollekes Live in concert	2021-04-27	410	2021-01-04	Snollebollekes Live in concert	2021-04-27
595	161	2021-02-02	Machi Music Festival	2020-01-25	434	2021-02-05	Machi Music Festival	2020-01-25
658	182	2021-06-18	Freshtival Weekend	2020-10-15	476	2021-06-19	Freshtival Weekend	2020-10-15
661	213	2021-05-19	Open'er Festival	2021-02-27	448	2021-05-20	Open'er Festival	2021-02-27

Interpretation of results: There are nine groups of matches that ended up swiping right / super swiped on each other and attended the same event. The "Snollebollekes Live in concert" has the most success. Bumble may want to organize this event again next year to increase the chance of people matching.

Query 11

Objective: To find if those with the premium plan receive more matches than those without.

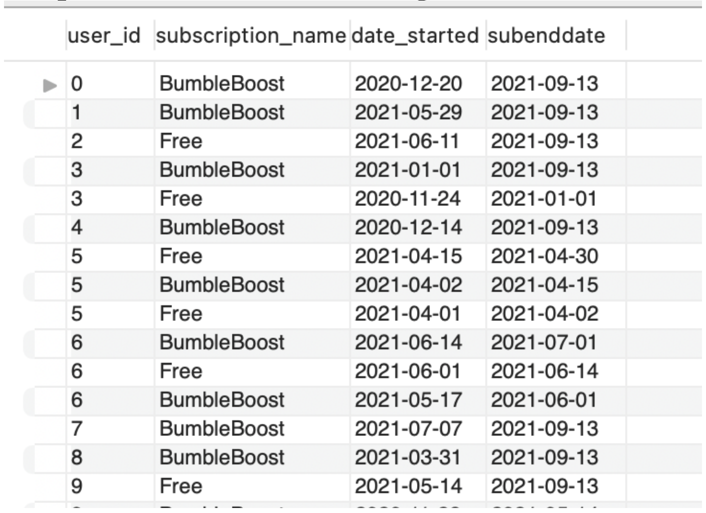
Assumptions: In case the premium plan ended (i.e. the user switched between plans), any match taking place after the end date won't be counted in the premium plan total and will be added to the total of matches received during the free plan. We modified the end_date of current plans to be today's date while creating the view table to simplify the query creation.

Code:

```
create view userssubscription as
select account_has_subscription.user_id, subscription.subscription_name, date_started,
case
        when account_has_subscription.date_ended is null then CURRENT_DATE
```

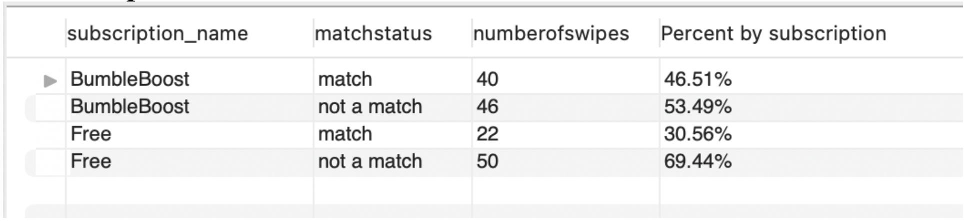
```
            else account_has_subscription.date_ended
end as subenddate
from subscription, account_has_subscription
where subscription.subscriptionID=account_has_subscription.subscriptionid order by user_id;

select subscription_name, matchstatus, count(*) as numberofswipes,
concat(round(count(*)/total*100,2), "%") as "Percent by subscription"
from matchstatus, userssubscription,

(Select count(*) as total
from matchstatus, userssubscription
where user_ID1=user_id and user_ID1date_seen>=date_started and
user_ID1date_seen<subenddate and subscription_name="BumbleBoost") as total

where user_ID1=user_id and user_ID1date_seen>=date_started and
user_ID1date_seen<subenddate and subscription_name="BumbleBoost"
group by subscription_name, matchstatus,total

union

select subscription_name, matchstatus, count(*) as numberofswipes,
concat(round(count(*)/total*100,2), "%") as "Percent by subscription"
from matchstatus, userssubscription,

(Select count(*) as total
from matchstatus, userssubscription
where user_ID1=user_id and user_ID1date_seen>=date_started and
user_ID1date_seen<subenddate and subscription_name="Free") as total

where user_ID1=user_id and user_ID1date_seen>=date_started and
user_ID1date_seen<subenddate and subscription_name="Free"

group by subscription_name, matchstatus,total;
```

Output from the 'userssubscription' view table:



user_id	subscription_name	date_started	subenddate
0	BumbleBoost	2020-12-20	2021-09-13
1	BumbleBoost	2021-05-29	2021-09-13
2	Free	2021-06-11	2021-09-13
3	BumbleBoost	2021-01-01	2021-09-13
3	Free	2020-11-24	2021-01-01
4	BumbleBoost	2020-12-14	2021-09-13
5	Free	2021-04-15	2021-04-30
5	BumbleBoost	2021-04-02	2021-04-15
5	Free	2021-04-01	2021-04-02
6	BumbleBoost	2021-06-14	2021-07-01
6	Free	2021-06-01	2021-06-14
6	BumbleBoost	2021-05-17	2021-06-01
7	BumbleBoost	2021-07-07	2021-09-13
8	BumbleBoost	2021-03-31	2021-09-13
9	Free	2021-05-14	2021-09-13

Final output:



subscription_name	matchstatus	numberofswipes	Percent by subscription
BumbleBoost	match	40	46.51%
BumbleBoost	not a match	46	53.49%
Free	match	22	30.56%
Free	not a match	50	69.44%

Interpretation of results: Users with the premium plan (i.e., BumbleBoost) received a match for 46.51% of their total swipes whereas for users using the app on a free subscription the swipe turned to a match only 30.56% of the time. Therefore, as a value proposition for the users looking to find 'love', the metric can be used to show that users should purchase BumbleBoost plan to increase their match rate by ~15%.

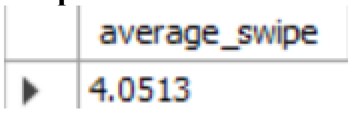
Query 12

Objective: To find the average number of swipes of our least active users before a successful match-up.

Assumptions: To be categorized as the least active users, the last active date should be greater than 2 months (from 2021/09/12)

Code:
```
SELECT total.swipe/success.suc AS success_rate
FROM (SELECT count(A.user_ID1) AS suc
FROM user_has_seen AS A, user_has_seen AS B, Account_and_login_info AS info, user_info
AS u
WHERE A.user_ID1=B.user_ID2
AND A.user_ID2=B.user_ID1
AND A.user_ID1 = u.user_ID
AND info.user_ID = u.user_ID
AND ((A.user_ID1_action=”right swipe” AND (B.user_ID1_action=”right swipe” OR
B.user_ID1_action=”super swipe”))
  OR
 (A.user_ID1_action=”super swipe” AND (B.user_ID1_action=”right swipe” OR
B.user_ID1_action=”super swipe”))
)
AND DATEDIFF(“2021-09-10”,info.last_active_date) > 60
    ) as success
    ,
    (SELECT count(A.user_ID1) as swipe
FROM user_has_seen AS A, Account_and_login_info AS info, user_info AS u
WHERE A.user_ID1=info.user_ID
AND info.user_ID = u.user_ID
    ) AS total;
```

Output:



	average_swipe
▶	4.0513

Interpretation of results: The results suggests that for those least-active users, on average there will be a successful match-up every 4 swipes, indicating a rather high successful rate. Bumble could use this information for promotion to attract more customers.

Query 13

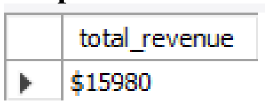
Objective: Calculate total revenue from subscriptions (all time revenue)

Assumptions: User will still pay for a monthly membership fee of $7.99 even if they cancel the service before the expire date of the premium plan. (Duration < 30 days).

Code:

```
SELECT CONCAT(“$”,ROUND(SUM(CEILING(DATEDIFF(
IFNULL(ahs.date_ended,’2021-09-12’),ahs.date_started)/30))*AVG(s.monthly_price),2))
AS total_revenue
FROM account_has_subscription AS ahs, Subscription AS s
WHERE ahs.subscriptionID = s.subscriptionID
AND ahs.subscriptionID = 102
```

Output:


	total_revenue
▶	$15980

Interpretation of results: The total revenue generated from last two years of operation of bumble shows the financial status of the company, which is profitable and could be used as a reference if they want to increase the profit by raising the price of the premium plan in the future.

Query 14

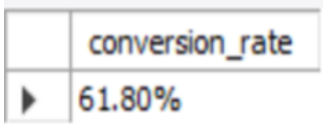
Objective: find the conversion rate of users changing from free subscription to paid subscription

Assumptions: A user who constantly switching from free plan and premium plan will only be considered as one successfully converted user.

Code:

```
SELECT CONCAT(ROUND(COUNT(DISTINCT sub.idNum)/
COUNT(DISTINCT ahs.user_ID)*100,2),"%")
AS conversion_rate
FROM account_has_subscription AS ahs,
(
SELECT ahs.user_ID as idNum
FROM account_has_subscription AS ahs
GROUP BY ahs.user_ID
HAVING SUM(ahs.subscriptionID=102)=1
) AS sub
```

Output:


	conversion_rate
▶	61.80%

Interpretation of results: The high conversion rate, which is higher than 50%, meaning more than half of our users are satisfied with our premium plan and are willing to pay for better services.

Theme 5 – User Targeting

Query 15

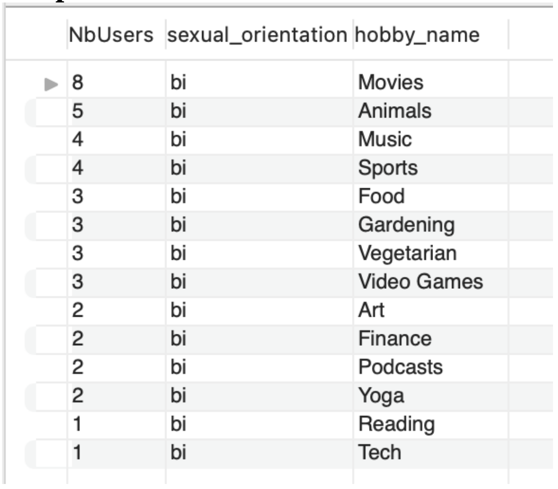
Objective: Find the most popular hobbies for a specific gender_oriented segment of our users. This would enable us to understand at a deeper level what may be useful content for that segment, thus offering more tailored suggestions of other users but as events (that would have more in common with their interest)

Assumptions: Most of our users are invited but not required to add hobbies (if they don't believe adding them will increase their compatibility for example). Therefore, we assume that we will only be looking at the users that did add hobbies to their account. This query uses the "USER_DEMOGRAPHICS" view table from query 18. ***Please run query 18 first before running query 15***.

Code:

```
/* Most Popular Hobby per Sexual Orientation */
SELECT count(distinct user_ID) as NbUsers,sexual_orientation, hobby_name
from USER_DEMOGRAPHICS
WHERE sexual_orientation = "bi"
GROUP BY sexual_orientation, hobby_name
ORDER BY NbUsers DESC;
```

Output:



NbUsers	sexual_orientation	hobby_name
8	bi	Movies
5	bi	Animals
4	bi	Music
4	bi	Sports
3	bi	Food
3	bi	Gardening
3	bi	Vegetarian
3	bi	Video Games
2	bi	Art
2	bi	Finance
2	bi	Podcasts
2	bi	Yoga
1	bi	Reading
1	bi	Tech

Interpretation of results: Here we selected a bisexual segment of our user base. We notice that the most popular hobby is "watching movies". This result can help us suggest or support more movie-based events on the app for example.

Query 16

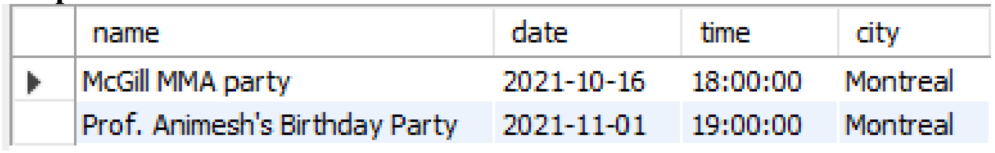
Objective: Find events that are relevant to the user: e.g., using his current location and the upcoming date of the event as selection criteria.

Assumptions: The user is required to enable their location to use the app as this is the only way we can efficiently suggest users that are physically meetable for the user. Therefore, we assume all users have authorised their location tracking and that their service is working. Moreover, premium plan users have the functionality to change their current location (to meet users from outside their town: for example, they are travelling to Montreal next week and they want to start chatting with people now) but this feature doesn't change the current location attribute in the user_info entity, it plays with another location attribute.

Code:

```
/* Recommend all the events happening in the same city as user */
/* All events, location and user who went */
CREATE or replace VIEW user_city as
SELECT u.user_id as user,u.first_name, u.last_name, l.city as "City"
from user_info as u
join location as l on u.current_latitude_longitude = l.latitude_longitude
GROUP BY user
ORDER BY user;
select * from user_city;

CREATE or replace VIEW event_city as
SELECT e.event_id as event,e.event_name as name, e.date, e.time, l.city as "City"
from event as e
join location as l on e.latitude_longitude = l.latitude_longitude
GROUP BY event
ORDER BY city;
select * from event_city;

Create or replace view user_suggested_events as
select ec.name, ec.date, ec.time, ec.city
from event_city as ec
INNER JOIN user_city as uc on uc.city = ec.city
WHERE uc.user = 25
and ec.date BETWEEN DATE(NOW()) and (DATE(NOW() + INTERVAL 2 MONTH))
ORDER BY date ASC, time ASC;

select * from user_suggested_events;
```

Output:



name	date	time	city
McGill MMA party	2021-10-16	18:00:00	Montreal
Prof. Animesh's Birthday Party	2021-11-01	19:00:00	Montreal

Interpretation of results: Here we see that the user has two events in his/her city coming up in the next two months. Therefore, we would use this result to feed the database of events we will be displaying to him/her.

Query 17

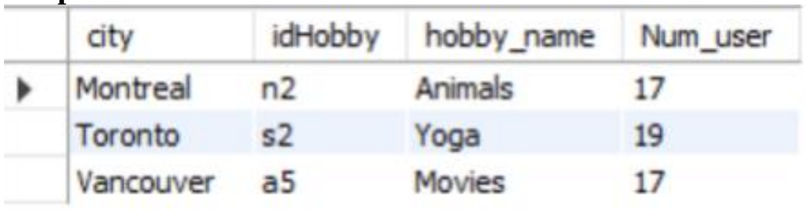
Objective: Find the most popular hobby in terms of locations.

Code:

```
CREATE or replace VIEW USER_city_hobby_montreal as
SELECT u.user_ID, l.city, uh.idHobby, h.hobby_name
FROM user_info as u
JOIN gender_orientation as go on u.idgender_orientation = go.idgender_orientation
JOIN user_info_has_Hobby as uh on u.user_ID = uh.user_ID
JOIN Hobby as h on uh.idHobby = h.idHobby
JOIN Location as l on u.current_latitude_longitude = l.latitude_longitude
WHERE l.city = "Montreal"
GROUP BY u.user_ID, uh.idHobby;

CREATE or replace VIEW USER_city_hobby_toronto as
SELECT u.user_ID, l.city, uh.idHobby, h.hobby_name
FROM user_info as u
JOIN gender_orientation as go on u.idgender_orientation = go.idgender_orientation
JOIN user_info_has_Hobby as uh on u.user_ID = uh.user_ID
JOIN Hobby as h on uh.idHobby = h.idHobby
JOIN Location as l on u.current_latitude_longitude = l.latitude_longitude
WHERE l.city = "Toronto"
GROUP BY u.user_ID, uh.idHobby;

CREATE or replace VIEW USER_city_hobby_vancouver as
SELECT u.user_ID, l.city, uh.idHobby, h.hobby_name
FROM user_info as u
JOIN gender_orientation as go on u.idgender_orientation = go.idgender_orientation
JOIN user_info_has_Hobby as uh on u.user_ID = uh.user_ID
JOIN Hobby as h on uh.idHobby = h.idHobby
JOIN Location as l on u.current_latitude_longitude = l.latitude_longitude
WHERE l.city = "Vancouver"
GROUP BY u.user_ID, uh.idHobby;

CREATE or replace VIEW m_fav as
SELECT m.city, m.idHobby, m.hobby_name, count(m.user_ID) as Num_user
from USER_city_hobby_montreal as m
group by m.idHobby
order by count(m.user_ID) DESC
LIMIT 1;

CREATE or replace VIEW v_fav as
SELECT v.city, v.idHobby, v.hobby_name, count(v.user_ID) as Num_user
from USER_city_hobby_vancouver as v
group by v.idHobby
```

```
order by count(v.user_ID) DESC
LIMIT 1;

CREATE or replace VIEW t_fav as
SELECT t.city, t.idHobby, t.hobby_name, count(t.user_ID) as Num_user
from USER_city_hobby_toronto as t
group by t.idHobby
order by count(t.user_ID) DESC
LIMIT 1;

SELECT * FROM m_fav
UNION
SELECT * FROM t_fav
UNION
SELECT * FROM v_fav
```

Output:



city	idHobby	hobby_name	Num_user
Montreal	n2	Animals	17
Toronto	s2	Yoga	19
Vancouver	a5	Movies	17

Interpretation of results: The result could be used for users to understand what is popular in their area and can be used as guidance for hosting different events in their areas.

Section 3 – Most Interesting and Complex Queries

Query 18

Logic behind the query / objective: To get a better understanding of the user demographics on the bumble platform and see the distribution of the different features recorded of our users segmented by age group and gender.

Challenges faced: Defining what are considered an interesting combinations and correlations (i.e., religion with politics or smoking and drinking) and defining the population of comparison range.

Overall learning outcomes: We learnt how to use various date functions to manipulate dates (timediff, currdate) and handling multiple view tables.

Assumptions:
We classified our users based the following structure:

- **16 to 18**: Teenagers
- **19 to 25**: Young Adults
- **26 to 39**: Adults
- **40 to 59**: Middle-aged Adults
- **60 to 79**: Old Adults
- **Over 80**: Seniors

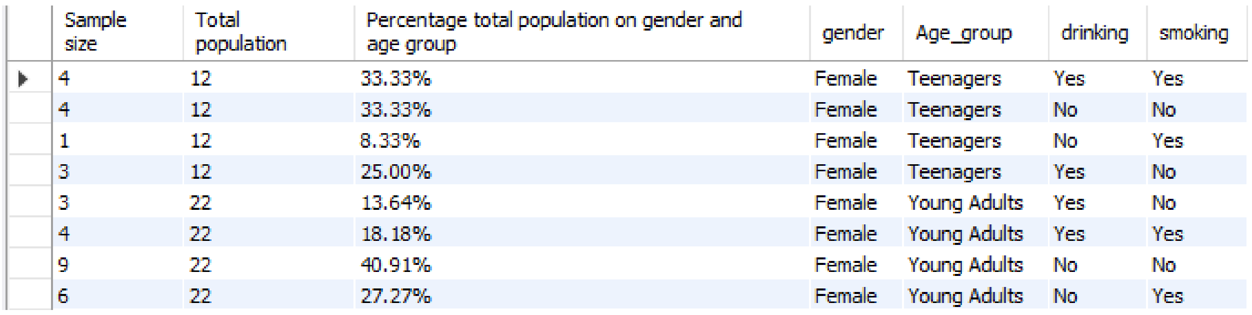
As you may notice there are no classification for users below 16 years old. We decided to classify those users as NA, as we would not authorise users under 16 to create a profile. If they did, we should flag them and exclude them from our reports.

Code:
```
/* Age Segmentation*/
CREATE or replace VIEW USER_DEMOGRAPHICS as
SELECT u.user_ID,(TIMESTAMPDIFF(YEAR, u.birthdate, CURDATE())) as Age,
case
        when (TIMESTAMPDIFF(YEAR, u.birthdate, CURDATE())) between 16 and 18 then
        "Teenagers"
        when (TIMESTAMPDIFF(YEAR, u.birthdate, CURDATE())) between 19 and 25 then
        "Young Adults"
        when (TIMESTAMPDIFF(YEAR, u.birthdate, CURDATE())) between 26 and 39 then
        "Adults"
        when (TIMESTAMPDIFF(YEAR, u.birthdate, CURDATE())) between 40 and 59 then
        "Middle-aged Adults"
        when (TIMESTAMPDIFF(YEAR, u.birthdate, CURDATE())) between 60 and 79 then
        "Old Adults"
        when (TIMESTAMPDIFF(YEAR, u.birthdate, CURDATE())) < 80 then "Seniors"
        else "NA"
end as Age_group,u.birthdate, u.zodiac, u.drinking, u.smoking, u.religion, u.politics,
go.gender, go.sexual_orientation, go.gender_interested_in,
h.hobby_name
FROM user_info as u
JOIN gender_orientation as go on u.idgender_orientation = go.idgender_orientation
JOIN user_info_has_Hobby as uh on u.user_ID = uh.user_ID
JOIN Hobby as h on uh.idHobby = h.idHobby
GROUP BY u.user_ID, h.hobby_name;

Create or replace view User_GA as
SELECT count(distinct user_ID) as NbUsers, gender, Age_group
from USER_DEMOGRAPHICS
GROUP BY gender, Age_group;

Create or replace view User_SD as
SELECT count(distinct user_ID) as NbUsers, gender, Age_group, drinking, smoking
from USER_DEMOGRAPHICS
GROUP BY gender, Age_group, drinking, smoking
ORDER BY CASE Age_group
   WHEN 'Teenagers' THEN 1
   WHEN 'Young Adults' THEN 2
   WHEN 'Adults' THEN 3
   WHEN 'Middle-aged Adults' THEN 4
   WHEN 'Old Adults' THEN 5
   WHEN 'Seniors' THEN 6
   ELSE 7
   END;

select User_SD.NBUsers as "Sample size", User_GA.NBUsers as "Total population",
concat(round(User_SD.NBUsers/User_GA.NBUsers*100,2),"%") "Percentage total population
on gender and age group", User_SD.gender, User_SD.Age_group, User_SD.drinking,
User_SD.smoking
from User_GA, User_SD
where User_GA.gender=User_SD.gender and  User_SD.Age_group=User_GA.Age_group
order by gender, CASE User_SD.Age_group
   WHEN 'Teenagers' THEN 1
   WHEN 'Young Adults' THEN 2
   WHEN 'Adults' THEN 3
   WHEN 'Middle-aged Adults' THEN 4
   WHEN 'Old Adults' THEN 5
   WHEN 'Seniors' THEN 6
   ELSE 7
   END;

Create or replace view User_RP as
SELECT count(distinct user_ID) as NbUsers, gender, Age_group, religion, politics
from USER_DEMOGRAPHICS
GROUP BY gender,Age_group, religion, politics;
```

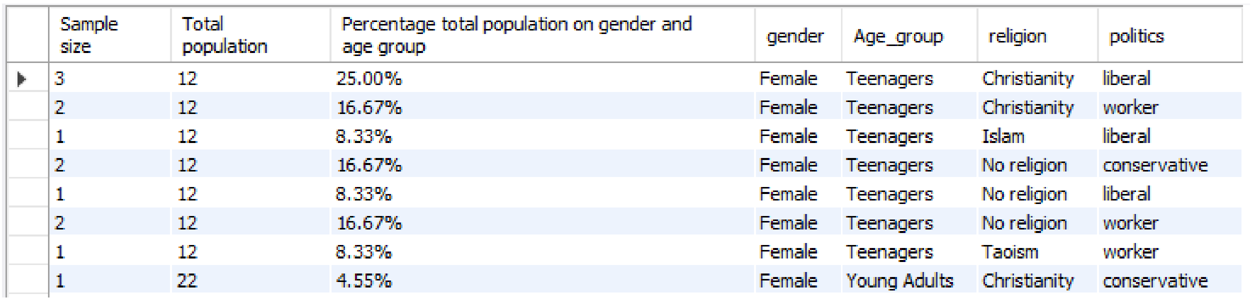
```
select User_RP.NBUsers as "Sample size", User_GA.NBUsers as "Total population",
concat(round(User_RP.NBUsers/User_GA.NBUsers*100,2),"%") "Percentage total population
on gender and age group", User_RP.gender, User_RP.Age_group,
User_RP.religion, User_RP.politics
from User_GA, User_RP
where User_GA.gender=User_RP.gender and User_RP.Age_group=User_GA.Age_group
GROUP BY gender, Age_group, religion, politics
order by gender, CASE User_RP.Age_group
   WHEN 'Teenagers' THEN 1
   WHEN 'Young Adults' THEN 2
   WHEN 'Adults' THEN 3
   WHEN 'Middle-aged Adults' THEN 4
   WHEN 'Old Adults' THEN 5
   WHEN 'Seniors' THEN 6
   ELSE 7
   END;

Create or replace view User_Orientation as
SELECT count(distinct user_ID) as NbUsers, Age_group, gender, sexual_orientation
from USER_DEMOGRAPHICS
GROUP BY Age_group, gender, sexual_orientation;

select User_Orientation.NBUsers as "Sample size", User_GA.NBUsers as "Total population",
concat(round(User_Orientation.NBUsers/User_GA.NBUsers*100,2),"%") "Percentage total
population on gender and age group", User_Orientation.gender, User_Orientation.Age_group,
User_Orientation.sexual_orientation
from User_GA, User_Orientation
where User_GA.gender=User_Orientation.gender and
User_Orientation.Age_group=User_GA.Age_group
GROUP BY gender, Age_group, sexual_orientation
order by gender, CASE User_Orientation.Age_group
   WHEN 'Teenagers' THEN 1
   WHEN 'Young Adults' THEN 2
   WHEN 'Adults' THEN 3
   WHEN 'Middle-aged Adults' THEN 4
   WHEN 'Old Adults' THEN 5
   WHEN 'Seniors' THEN 6
   ELSE 7
   END;

Create or replace view User_hobbies as
SELECT count(distinct user_ID) as NbUsers, Age_group, gender, hobby_name
from USER_DEMOGRAPHICS
GROUP BY Age_group, gender, hobby_name;
```

```
select User_hobbies.NBUsers as "Sample size", User_GA.NBUsers as "Total population",
concat(round(User_hobbies.NBUsers/User_GA.NBUsers*100,2),"%") "Percentage total
population on gender and age group", User_hobbies.gender, User_hobbies.Age_group,
User_hobbies.hobby_name as hobby
from User_GA, User_hobbies
where User_GA.gender=User_hobbies.gender and
User_hobbies.Age_group=User_GA.Age_group
GROUP BY gender, Age_group, hobby
order by gender, CASE User_hobbies.Age_group
   WHEN 'Teenagers' THEN 1
   WHEN 'Young Adults' THEN 2
   WHEN 'Adults' THEN 3
   WHEN 'Middle-aged Adults' THEN 4
   WHEN 'Old Adults' THEN 5
   WHEN 'Seniors' THEN 6
   ELSE 7
   END;
```

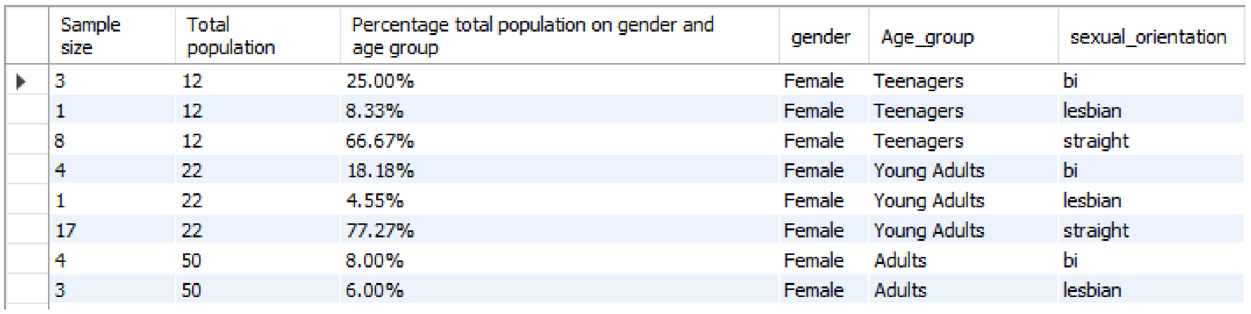
Output:



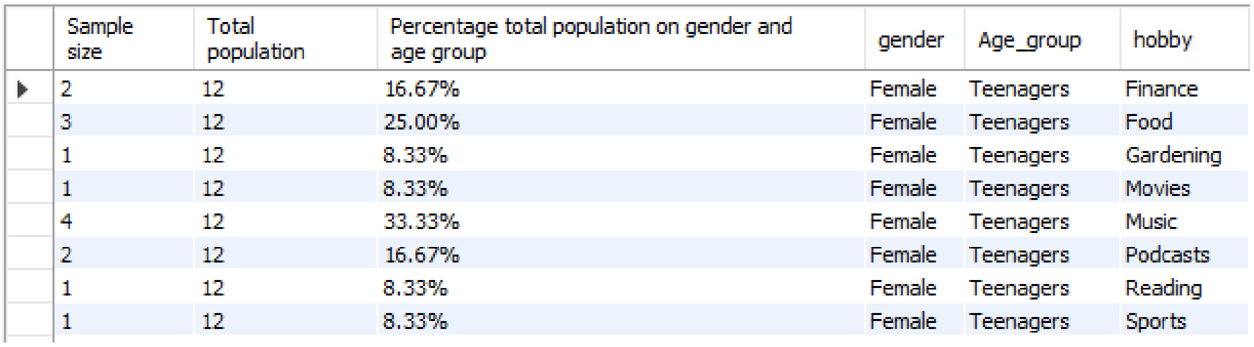
Sample size	Total population	Percentage total population on gender and age group	gender	Age_group	drinking	smoking
4	12	33.33%	Female	Teenagers	Yes	Yes
4	12	33.33%	Female	Teenagers	No	No
1	12	8.33%	Female	Teenagers	No	Yes
3	12	25.00%	Female	Teenagers	Yes	No
3	22	13.64%	Female	Young Adults	Yes	No
4	22	18.18%	Female	Young Adults	Yes	Yes
9	22	40.91%	Female	Young Adults	No	No
6	22	27.27%	Female	Young Adults	No	Yes



Sample size	Total population	Percentage total population on gender and age group	gender	Age_group	religion	politics
3	12	25.00%	Female	Teenagers	Christianity	liberal
2	12	16.67%	Female	Teenagers	Christianity	worker
1	12	8.33%	Female	Teenagers	Islam	liberal
2	12	16.67%	Female	Teenagers	No religion	conservative
1	12	8.33%	Female	Teenagers	No religion	liberal
2	12	16.67%	Female	Teenagers	No religion	worker
1	12	8.33%	Female	Teenagers	Taoism	worker
1	22	4.55%	Female	Young Adults	Christianity	conservative



Sample size	Total population	Percentage total population on gender and age group	gender	Age_group	sexual_orientation
3	12	25.00%	Female	Teenagers	bi
1	12	8.33%	Female	Teenagers	lesbian
8	12	66.67%	Female	Teenagers	straight
4	22	18.18%	Female	Young Adults	bi
1	22	4.55%	Female	Young Adults	lesbian
17	22	77.27%	Female	Young Adults	straight
4	50	8.00%	Female	Adults	bi
3	50	6.00%	Female	Adults	lesbian



Sample size	Total population	Percentage total population on gender and age group	gender	Age_group	hobby
2	12	16.67%	Female	Teenagers	Finance
3	12	25.00%	Female	Teenagers	Food
1	12	8.33%	Female	Teenagers	Gardening
1	12	8.33%	Female	Teenagers	Movies
4	12	33.33%	Female	Teenagers	Music
2	12	16.67%	Female	Teenagers	Podcasts
1	12	8.33%	Female	Teenagers	Reading
1	12	8.33%	Female	Teenagers	Sports

Interpretation of results: We see here that we have an interesting distribution of users across each feature. Equally this query helps us have a constant view on our studies since we are ordering the results in the same manner starting with the youngest female age group.

Query 19

Logic behind the query / objective: The objective of the query was to find matches that have the most characteristics in common with each other and assess them based on compatibility. This will help Bumble, see if the algorithm is working and producing successful matches.

Challenges faced: There were many steps needed to break down the query. We needed to apply multiple unions and "dummy variable encoding" to help us assess how compatible the two matches were with each other.

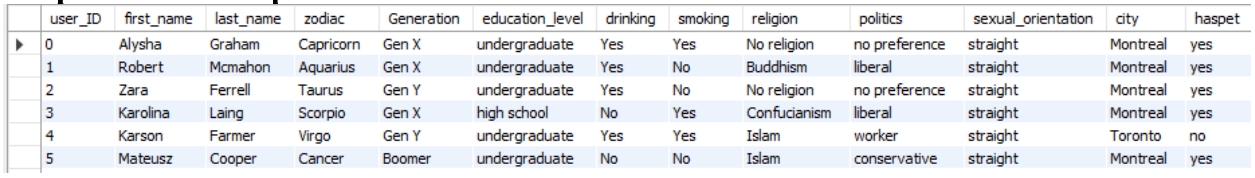
Overall learning outcomes: We learnt how to apply case functions, handling of null values, and applying dummy variable encoding to calculate the percentage of similarity between matched users.

Assumptions: We classified each user into their respective generation categories based on the following link: https://dailyfreepress.com/2021/03/15/generation-names-explained/. The matches view table from query 8 is used.

Code:
```
/*updated userinfo table*/
create view updateduserinfo as
select user_info.user_ID, first_name, last_name, zodiac,
case
        when year(birthdate)<=1964 then "Boomer"
        when year(birthdate)<=1980 then "Gen X"
        wen year(birthdate)<=1996 then "Gen Y"
        else "Gen Z"
end as "Generation", education_level, drinking, smoking, religion, politics, sexual_orientation,
city, haspet
from user_info, location, gender_orientation,
(select distinct user_id,
case
	when Quantity is null then "no"
	else "yes"
end as haspet
from user_info_has_pet) as userhaspet
where user_info.current_latitude_longitude=location.latitude_longitude and
user_info.idgender_orientation=gender_orientation.idgender_orientation and
userhaspet.user_id=user_info.user_id;


/* create binary coding*/
create view sameuserinterest as
select user_ID1, concat(A.first_name, " ", A.last_name) as user_ID1_Name, user_ID2,
concat(B.first_name, " ", B.last_name) as user_ID2_Name,
case
	when A.zodiac=B.zodiac then 1
	else 0
end as samezodiac,

case
	when A.Generation=B.Generation then 1
	else 0
end as samegeneration,

case
	when A.education_level=B.education_level then 1
	else 0
end as sameeducation,

case
	when A.drinking=B.drinking then 1
	else 0
end as samedrinking,

case
	when A.smoking=B.smoking then 1
	else 0
end as samesmoking,

case
	when A.religion=B.religion then 1
	else 0
end as samereligion,

case
	when A.politics=B.politics then 1
```

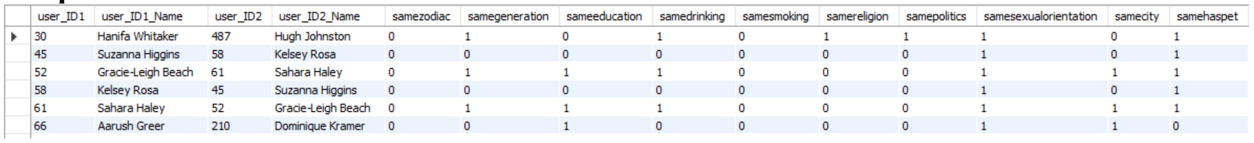
```
        else 0
end as samepolitics,

case
        when A.sexual_orientation=B.sexual_orientation then 1
        else 0
end as samesexualorientation,

case
        when A.city=B.city then 1
        else 0
end as samecity,

case
        when A.haspet=B.haspet then 1
        else 0
end as samehaspet

from
(select user_ID1, user_ID2 from matchstatus where matchstatus="match") as
matches,updateduserinfo as A, updateduserinfo as B
where matches.user_ID1=A.user_id and matches.user_ID2=B.user_ID;

/* display the compatibility score of the matched individuals*/
select user_ID1+user_ID2 as matchID, user_ID1, user_ID1_Name, user_ID2, user_ID2_Name,
concat(round((samezodiac+samegeneration+sameeducation+samedrinking+samesmoking+samer
eligion+samepolitics+samesexualorientation+samecity+samehaspet)/10*100,0),"%") as
"Compatibility Score"
from sameuserinterest
group by matchID
order by
(samezodiac+samegeneration+sameeducation+samedrinking+samesmoking+samereligion+same
politics+samesexualorientation+samecity+samehaspet)/10*100 desc;
```

Output from the "updateduserinfo" view table:



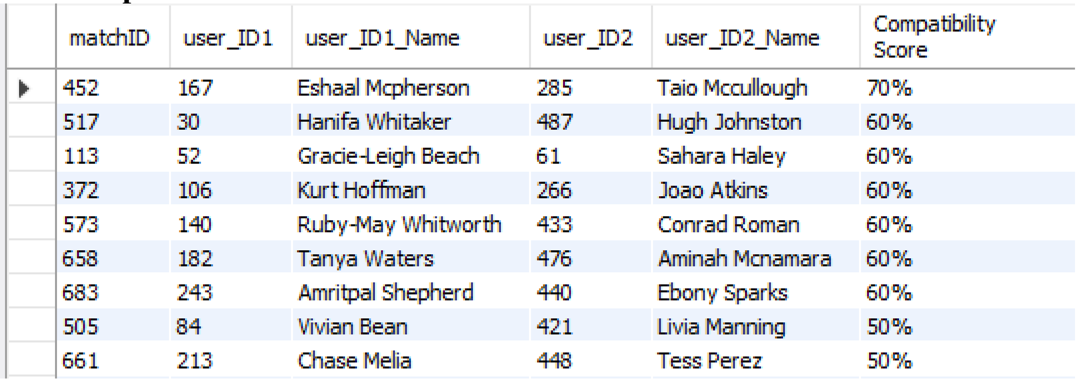
user_ID	first_name	last_name	zodiac	Generation	education_level	drinking	smoking	religion	politics	sexual_orientation	city	haspet
0	Alysha	Graham	Capricorn	Gen X	undergraduate	Yes	Yes	No religion	no preference	straight	Montreal	yes
1	Robert	Mcmahon	Aquarius	Gen X	undergraduate	Yes	No	Buddhism	liberal	straight	Montreal	yes
2	Zara	Ferrell	Taurus	Gen Y	undergraduate	Yes	No	No religion	no preference	straight	Montreal	yes
3	Karolina	Laing	Scorpio	Gen X	high school	No	Yes	Confucianism	liberal	straight	Montreal	yes
4	Karson	Farmer	Virgo	Gen Y	undergraduate	Yes	Yes	Islam	worker	straight	Toronto	no
5	Mateusz	Cooper	Cancer	Boomer	undergraduate	No	No	Islam	conservative	straight	Montreal	yes

Output from the "sameuserinterest" view table:



user_ID1	user_ID1_Name	user_ID2	user_ID2_Name	samezodiac	samegeneration	sameeducation	samedrinking	samesmoking	samereligion	samepolitics	samesexualorientation	samecity	samehaspet
30	Hanifa Whitaker	487	Hugh Johnston	0	1	0	1	0	1	1	1	0	1
45	Suzanna Higgins	58	Kelsey Rosa	0	0	0	0	0	0	0	1	0	1
52	Gracie-Leigh Beach	61	Sahara Haley	0	1	1	1	0	0	0	1	1	1
58	Kelsey Rosa	45	Suzanna Higgins	0	0	0	0	0	0	0	1	0	1
61	Sahara Haley	52	Gracie-Leigh Beach	0	1	1	1	0	0	0	1	1	1
66	Aarush Greer	210	Dominique Kramer	0	0	1	0	0	0	0	1	1	0

Final output:



matchID	user_ID1	user_ID1_Name	user_ID2	user_ID2_Name	Compatibility Score
452	167	Eshaal Mcpherson	285	Taio Mccullough	70%
517	30	Hanifa Whitaker	487	Hugh Johnston	60%
113	52	Gracie-Leigh Beach	61	Sahara Haley	60%
372	106	Kurt Hoffman	266	Joao Atkins	60%
573	140	Ruby-May Whitworth	433	Conrad Roman	60%
658	182	Tanya Waters	476	Aminah Mcnamara	60%
683	243	Amritpal Shepherd	440	Ebony Sparks	60%
505	84	Vivian Bean	421	Livia Manning	50%
661	213	Chase Melia	448	Tess Perez	50%

Interpretation of results: The most compatible pair is Eshaal and Taio with a compatibility score of 70%. This is the highest score within the list of matches. Approximately 1/3 of the matches have a compatibility score of 50% or higher. Bumble should improve its algorithm to ensure that people with the same specifications are shown to each other more.

Query 20 - New feature

Logic behind the query: Provide a new feature on the app, like Facebook's badges, to give users their own badges that describe their characteristics and "achievements". This will help with "customer loyalty" because it acknowledges the user's unique achievements.

Challenges faced: Having to combine different queries and logic into a final table that could be implemented with the app as a new feature.

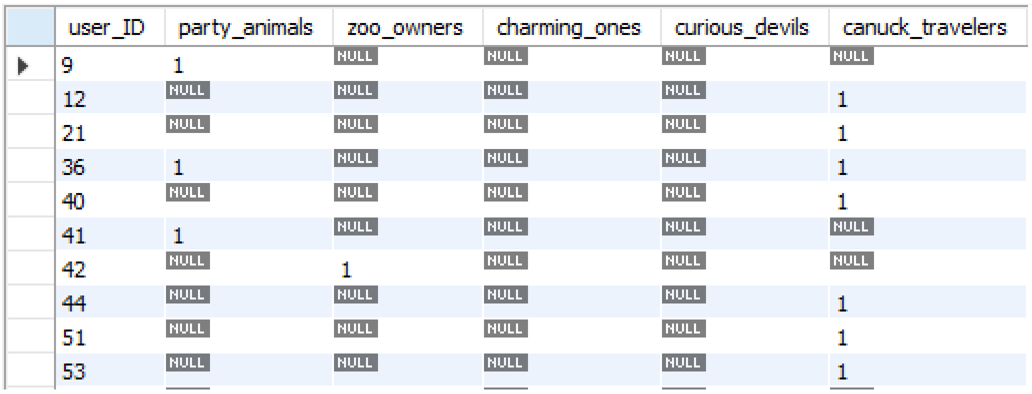
Assumptions:

- "party animals" badge is given to the users who attended equal to or more than 3 events;
- "zoo owners" badge is given to the users who have equal to or more than 5 pets;
- "charming ones" badge is given to the users who were right or super swiped by others for equal to or more than twice;
- "curious devils" badge is given to the users who made equal to or more than 3 swipes;
- "canuck travelers" bacge is given to the users who have multiple city information in their profiles.

Code:

```
/* create another table: badge */
CREATE TABLE badge (
user_ID INT);

/* insert user_ID as primary key */
INSERT INTO badge
SELECT user_ID FROM user_info;

ALTER TABLE badge
CHANGE COLUMN user_ID user_ID INT NOT NULL,
ADD PRIMARY KEY (user_ID);

/* add columns to have each column represent one kind of badge */
ALTER TABLE badge
        ADD COLUMN party_animals varchar(255) DEFAULT NULL,
        ADD COLUMN zoo_owners varchar(255) DEFAULT NULL,
        ADD COLUMN charming_ones varchar(255) DEFAULT NULL,
        ADD COLUMN curious_devils varchar(255) DEFAULT NULL,
        ADD COLUMN canuck_travelers varchar(255) DEFAULT NULL;
SET SQL_SAFE_UPDATES = 0;

/* reward users with badges based on their characteristics and achievements */
UPDATE badge
SET party_animals = 1
WHERE user_ID IN (SELECT sub.user_ID
FROM (
SELECT user_ID, COUNT(event_id) AS events_attended
FROM user_info_has_event
GROUP BY user_ID
) AS sub
  WHERE sub.events_attended >= 3);
UPDATE badge
SET zoo_owners = 1
WHERE user_ID IN (SELECT sub.user_ID AS zoo_owners
FROM (
SELECT user_ID, SUM(quantity) AS pets_owned
FROM user_info_has_pet
GROUP BY user_ID
) AS sub
WHERE sub.pets_owned >= 5);

UPDATE badge
SET charming_ones = 1
WHERE user_ID IN (SELECT sub.user_ID2
FROM (
SELECT user_ID2, COUNT(user_ID2) AS swiped_count
FROM user_info, user_has_seen, gender_orientation
WHERE (user_has_seen.user_ID1_action = "right swipe" OR user_has_seen.user_ID1_action =
"super_swipe")
AND user_info.user_ID=user_has_seen.user_ID2
AND gender_orientation.idgender_orientation=user_info.idgender_orientation
GROUP BY user_ID2
) AS sub
WHERE sub.swiped_count > 1);

UPDATE badge
SET curious_devils = 1
WHERE user_ID IN (SELECT sub.USER_ID1
FROM (
SELECT user_ID1, COUNT(user_ID1) AS swipes_made
FROM user_has_seen
GROUP BY user_ID1
) AS sub
WHERE sub.swipes_made >=3);

UPDATE badge
SET canuck_travelers = 1
WHERE user_ID IN (SELECT sub.ids
FROM (
SELECT u.user_ID AS ids, COUNT(u.user_ID) AS occurence
FROM location AS l
INNER JOIN premium_plan_locations AS pre ON l.latitude_longitude = pre.latitude_longitude
INNER JOIN user_info AS u ON pre.user_ID = u.user_Id
GROUP BY u.user_ID
) AS sub
WHERE sub.occurence >=2);

/* Show the resulting users with badges*/
SELECT *
FROM badge
where 1 in (party_animals, zoo_owners,charming_ones,curious_devils,canuck_travelers);
```

Output:



user_ID	party_animals	zoo_owners	charming_ones	curious_devils	canuck_travelers
9	1	NULL	NULL	NULL	NULL
12	NULL	NULL	NULL	NULL	1
21	NULL	NULL	NULL	NULL	1
36	1	NULL	NULL	NULL	1
40	NULL	NULL	NULL	NULL	1
41	1	NULL	NULL	NULL	NULL
42	NULL	1	NULL	NULL	NULL
44	NULL	NULL	NULL	NULL	1
51	NULL	NULL	NULL	NULL	1
53	NULL	NULL	NULL	NULL	1

Interpretation of results: The badges were given to the users who fulfill the requirement of specific characters or achievements. Further user incentive activities, advertising and badge-based data analysis can be undertaken in terms of this new feature.