G60:
Kexin Fang, 260773389
Pinghsuan Chueh, 260709676
Xiaohui Wang, 260719359
Zhenghua Chen, 260783959
Password: comp421___
(There are three “_”s)

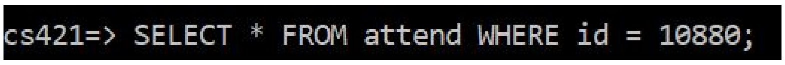
COMP421 Project 3

Question 1:

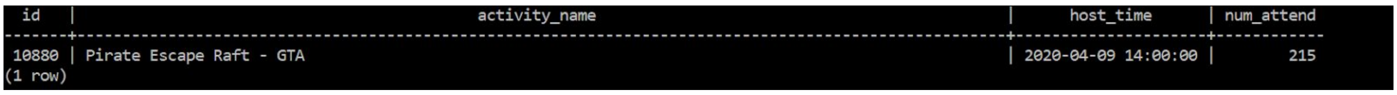
Idea: by this procedure we recommend a list of activities that the user might be interested in according to the recent games the user or his/her friends are playing; we also reflect the changes in relation when the user is so boring and agrees to take part in all of these recommended activities.

After inputting a user ID and a year, check the games that the user or his/hers friends played since that year (by default since the date ‘yearIn-01-01’). Then find all the activities based on these games which started after that year.
Output a relation of such activities, displaying activity name and host time, game id, id of a recently-played friend (or user himself/herself).
Then for the input user to take part in those activities, insert additional tuples into ATTEND relation indicating input user’s id, activity name, host time. (attend_num would be in another stored procedure to count the number of participants by group of activity to calculate, here for simplicity we just put NULL in that field)

Before the execution:



```
cs421=> SELECT * FROM attend WHERE id = 10880;
```

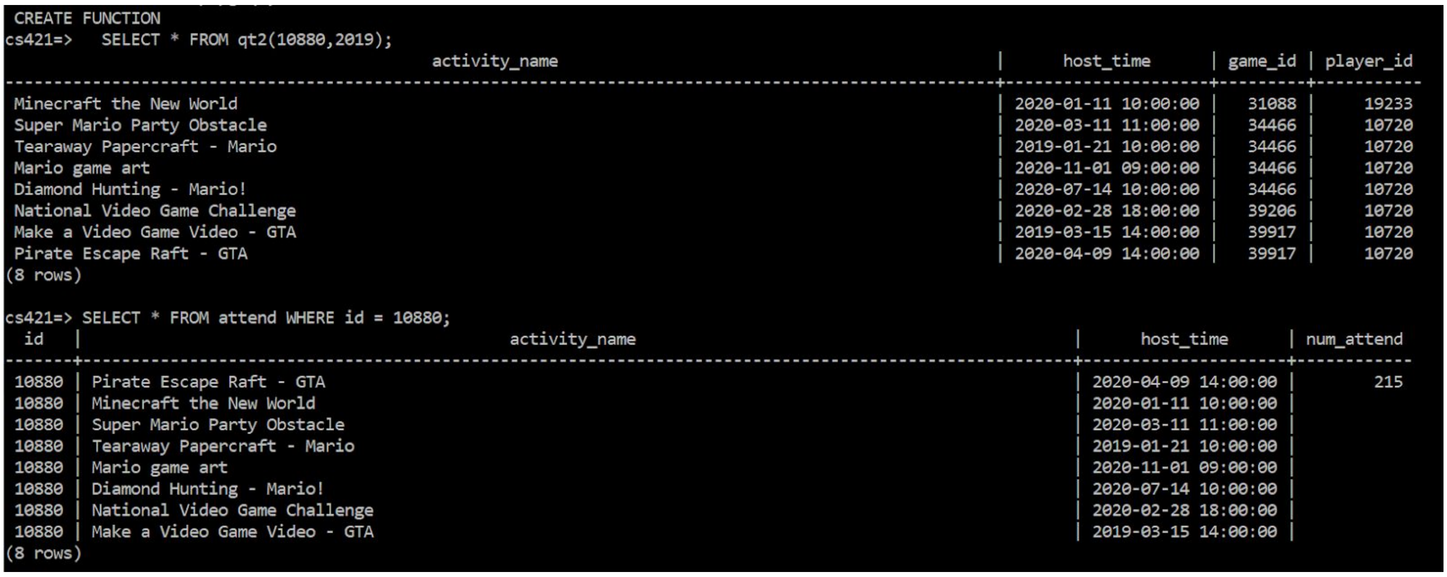


```
  id   |                              activity_name                              |      host_time      | num_attend
-------+-------------------------------------------------------------------------+---------------------+------------
 10880 | Pirate Escape Raft - GTA                                                | 2020-04-09 14:00:00 |        215
(1 row)
```

Stored Procedure:

```
CREATE OR REPLACE FUNCTION qt2(userID int, yearIn int) RETURNS
TABLE(activity_name char(100),host_time timestamp, game_id int, player_id int) AS
$$
 DECLARE interestDate date = make_date(yearIn, 1, 1);
 DECLARE interestGame RECORD;
 DECLARE interestActivity RECORD;
 DECLARE C1 CURSOR FOR
  SELECT  play.game_id AS gameID, min(play.id) AS playerID FROM play JOIN (
  (SELECT DISTINCT user1 AS id FROM friend WHERE user1 = userID OR user2
= userID) UNION
  (SELECT DISTINCT user2 AS id FROM friend WHERE user1 = userID OR user2
= userID)) AS playerSet
  ON play.id = playerSet.id AND late_play_date>interestDate
  GROUP BY play.game_id;
 DECLARE C2 CURSOR (game int) FOR
  SELECT host.activity_name, host.host_time, host.game_id FROM host WHERE
host.game_id=game AND host.host_time>interestDate;
BEGIN
   FOR interestGame IN C1 LOOP
     FOR interestActivity IN C2(interestGame.gameID) LOOP
          IF NOT EXISTS (SELECT 1 FROM attend WHERE attend.id=userID
                         AND attend.activity_name=interestActivity.activity_name)
          THEN
          INSERT INTO attend VALUES
   (userID,interestActivity.activity_name,interestActivity.host_time,NULL);
       END IF;
    activity_name :=interestActivity.activity_name;
    host_time := interestActivity.host_time;
    game_id := interestActivity.game_id;
    player_id := interestGame.playerID;
    RETURN NEXT;
    END LOOP;
  END LOOP;
END;
$$ LANGUAGE plpgsql;
```

Execute the procedure:



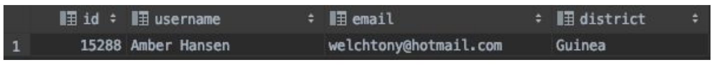
```
 CREATE FUNCTION
cs421=>   SELECT * FROM qt2(10880,2019);
                                  activity_name                                    |      host_time      | game_id | player_id
-----------------------------------------------------------------------------------+---------------------+---------+-----------
 Minecraft the New World                                                           | 2020-01-11 10:00:00 |   31088 |     19233
 Super Mario Party Obstacle                                                        | 2020-03-11 11:00:00 |   34466 |     10720
 Tearaway Papercraft - Mario                                                       | 2019-01-21 10:00:00 |   34466 |     10720
 Mario game art                                                                    | 2020-11-01 09:00:00 |   34466 |     10720
 Diamond Hunting - Mario!                                                          | 2020-07-14 10:00:00 |   34466 |     10720
 National Video Game Challenge                                                     | 2020-02-28 18:00:00 |   39206 |     10720
 Make a Video Game Video - GTA                                                     | 2019-03-15 14:00:00 |   39917 |     10720
 Pirate Escape Raft - GTA                                                          | 2020-04-09 14:00:00 |   39917 |     10720
(8 rows)

cs421=> SELECT * FROM attend WHERE id = 10880;
  id   |                                  activity_name                                    |      host_time      | num_attend
-------+-----------------------------------------------------------------------------------+---------------------+------------
 10880 | Pirate Escape Raft - GTA                                                          | 2020-04-09 14:00:00 |        215
 10880 | Minecraft the New World                                                           | 2020-01-11 10:00:00 |
 10880 | Super Mario Party Obstacle                                                        | 2020-03-11 11:00:00 |
 10880 | Tearaway Papercraft - Mario                                                       | 2019-01-21 10:00:00 |
 10880 | Mario game art                                                                    | 2020-11-01 09:00:00 |
 10880 | Diamond Hunting - Mario!                                                          | 2020-07-14 10:00:00 |
 10880 | National Video Game Challenge                                                     | 2020-02-28 18:00:00 |
 10880 | Make a Video Game Video - GTA                                                     | 2019-03-15 14:00:00 |
(8 rows)
```

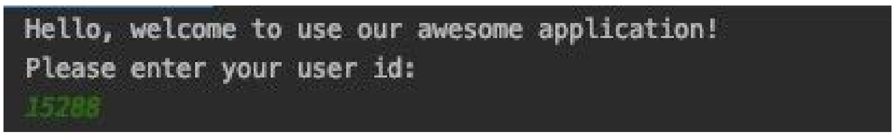
Question 2: Write user-friendly application program in Java

We made six alternative options for our program. Each is presented a demonstration showing the program running, and brief explanations of the alternative. The code could be found in application.java; exception and error handling is integrated in the application.

SQL codes to be implemented in Java:
The client using our application has the user id: 15288.



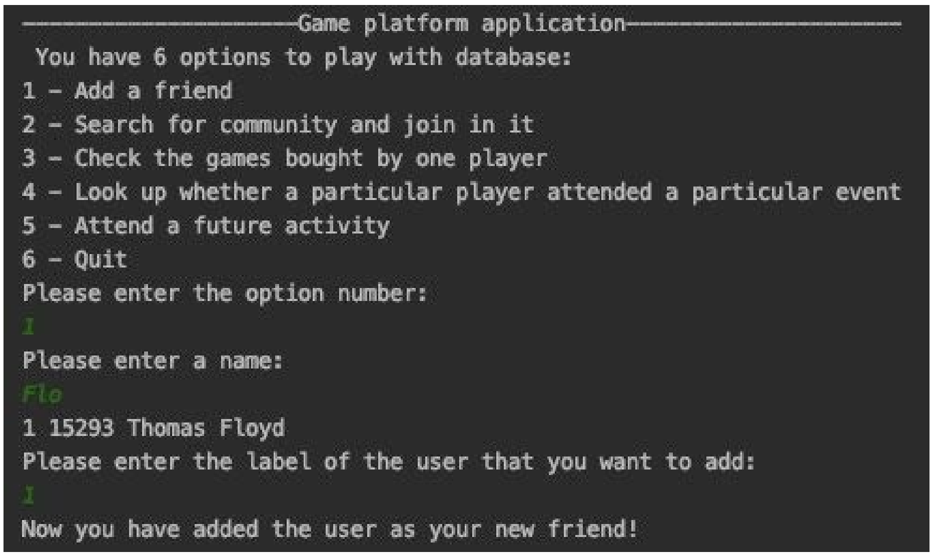
	id	username	email	district
1	15288	Amber Hansen	welchtony@hotmail.com	Guinea



```
Hello, welcome to use our awesome application!
Please enter your user id:
15288
```

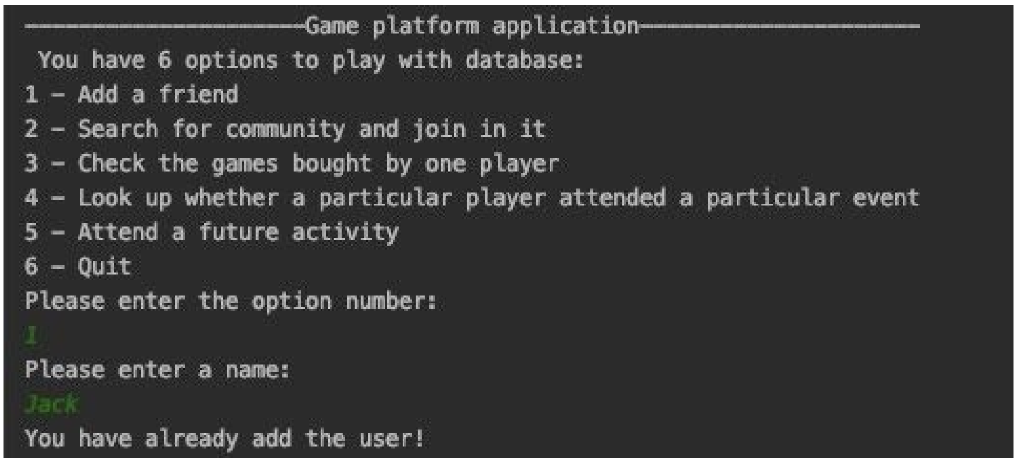
1. **Search for user by name, and select user from search result to add friend, if not friended yet**
   Part 1: user input user name (substring is fine)
   Part 2: user input the label of the user to be friended with

   Application demonstration:
   Case 1 - selected player is added to friend list



```
------------------------Game platform application------------------------
 You have 6 options to play with database:
1 - Add a friend
2 - Search for community and join in it
3 - Check the games bought by one player
4 - Look up whether a particular player attended a particular event
5 - Attend a future activity
6 - Quit
Please enter the option number:
1
Please enter a name:
Flo
1 15293 Thomas Floyd
Please enter the label of the user that you want to add:
1
Now you have added the user as your new friend!
```

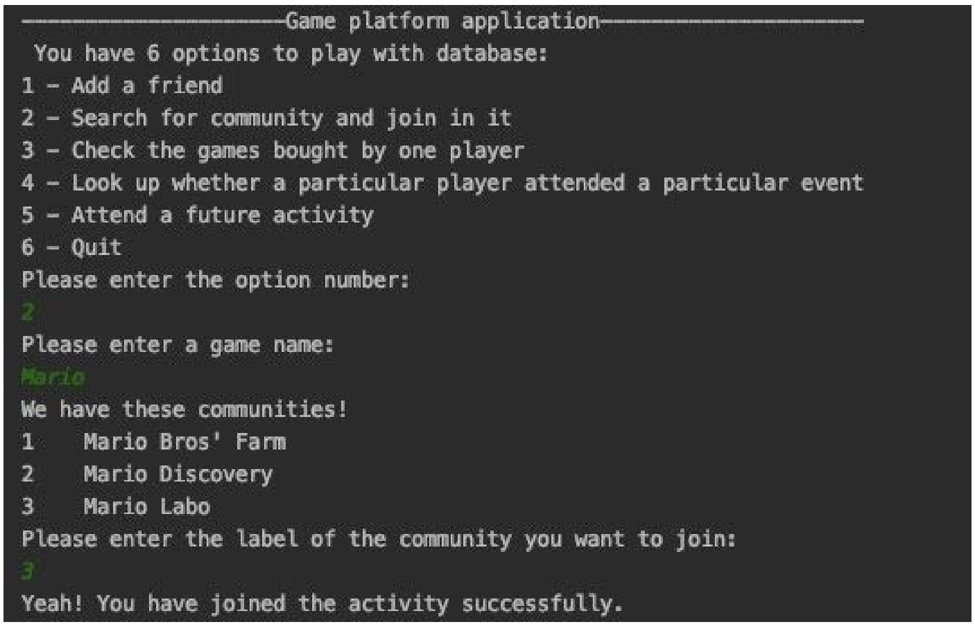
   Case 2 - selected player is already on player’s friend list



```
---------------------Game platform application---------------------
 You have 6 options to play with database:
1 - Add a friend
2 - Search for community and join in it
3 - Check the games bought by one player
4 - Look up whether a particular player attended a particular event
5 - Attend a future activity
6 - Quit
Please enter the option number:
1
Please enter a name:
Jack
You have already add the user!
```

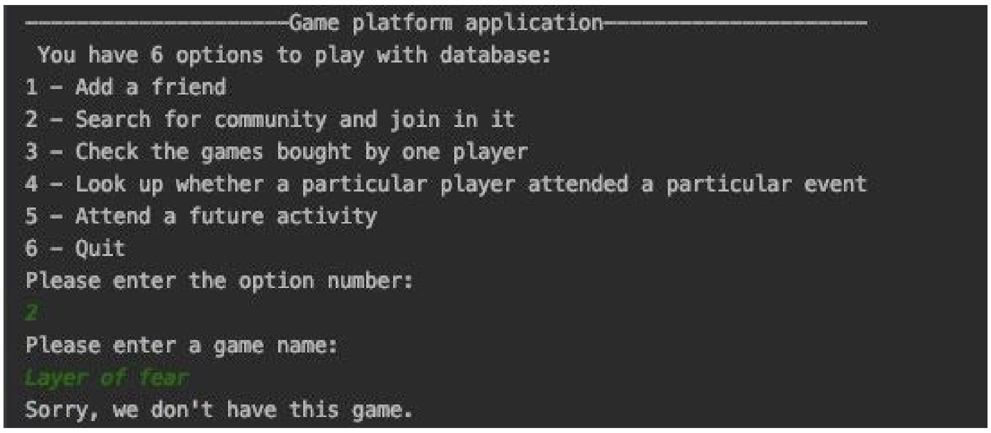
2. **Search for community by game name and select a community to join in**
   Part 1: user input game name
   Part 2: user input the label of the community that he/she wants to join in

   Application demonstration:
   Case 1 - client joins in the selected community



```
---------------------Game platform application---------------------
 You have 6 options to play with database:
1 - Add a friend
2 - Search for community and join in it
3 - Check the games bought by one player
4 - Look up whether a particular player attended a particular event
5 - Attend a future activity
6 - Quit
Please enter the option number:
2
Please enter a game name:
Mario
We have these communities!
1    Mario Bros' Farm
2    Mario Discovery
3    Mario Labo
Please enter the label of the community you want to join:
3
Yeah! You have joined the activity successfully.
```

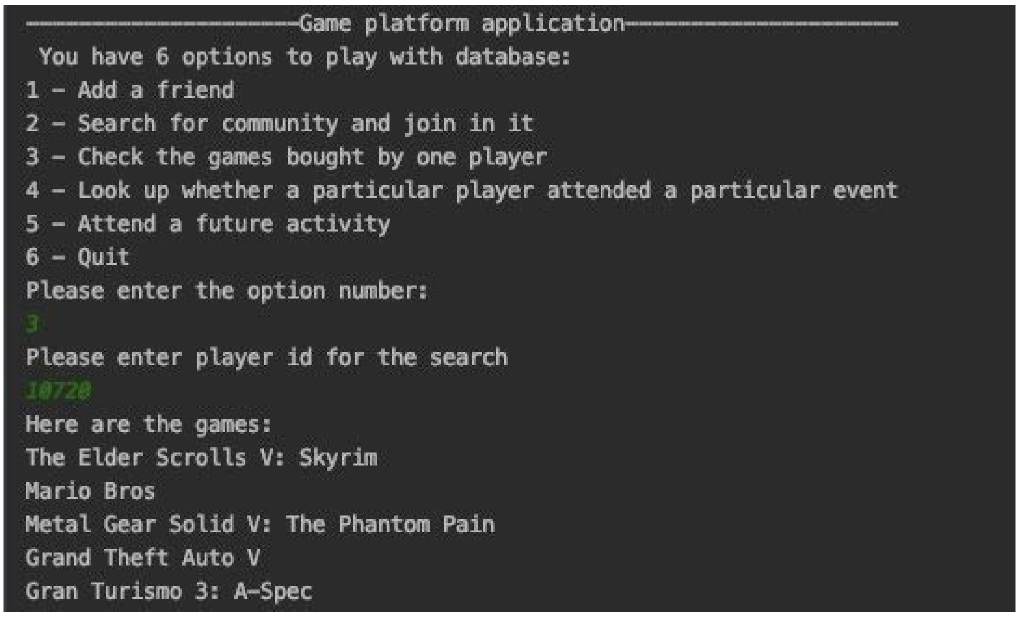
   Case 2 - client entered invalid search prompt (e.g. the game doesn’t exist)



```
----------------------Game platform application----------------------
 You have 6 options to play with database:
1 - Add a friend
2 - Search for community and join in it
3 - Check the games bought by one player
4 - Look up whether a particular player attended a particular event
5 - Attend a future activity
6 - Quit
Please enter the option number:
2
Please enter a game name:
Layer of fear
Sorry, we don't have this game.
```

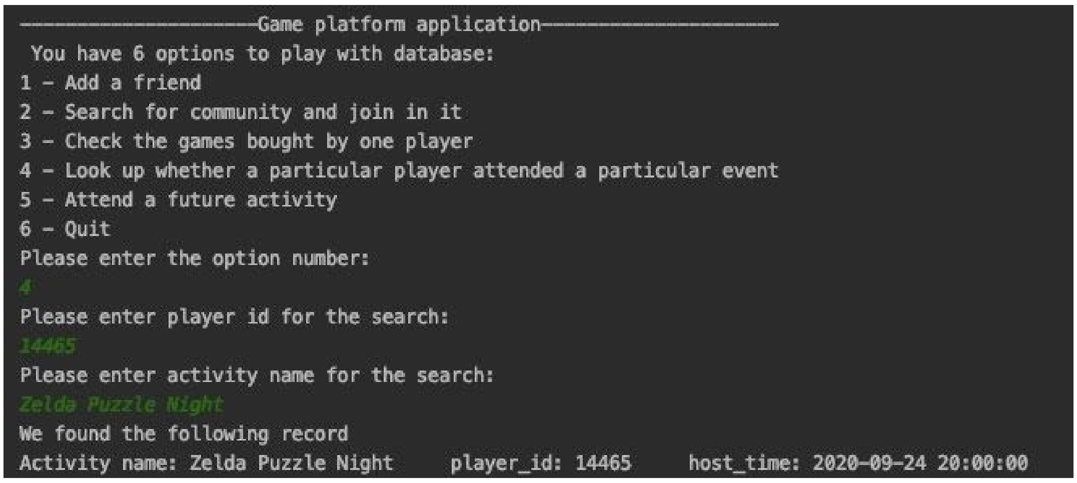
3. **Enter player id and display list of games the player has bought**
   User input player id

   Application demonstration:



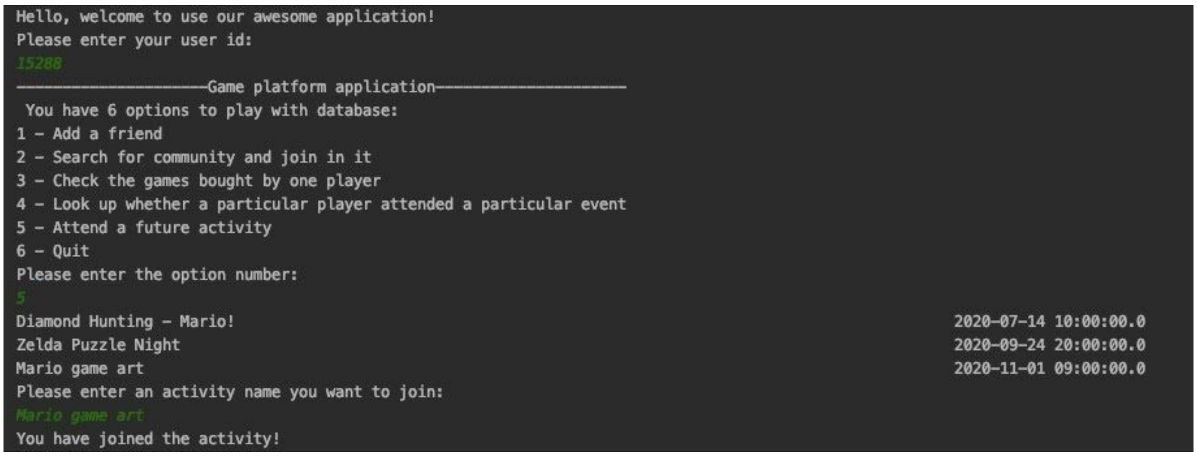
```
----------------------Game platform application----------------------
 You have 6 options to play with database:
1 - Add a friend
2 - Search for community and join in it
3 - Check the games bought by one player
4 - Look up whether a particular player attended a particular event
5 - Attend a future activity
6 - Quit
Please enter the option number:
3
Please enter player id for the search
10720
Here are the games:
The Elder Scrolls V: Skyrim
Mario Bros
Metal Gear Solid V: The Phantom Pain
Grand Theft Auto V
Gran Turismo 3: A-Spec
```

4. **Look up whether a particular player attend a particular activity**
   User input player id, then user input activity name
   If the player attended the activity, the application will print the record.
   Application demonstration:

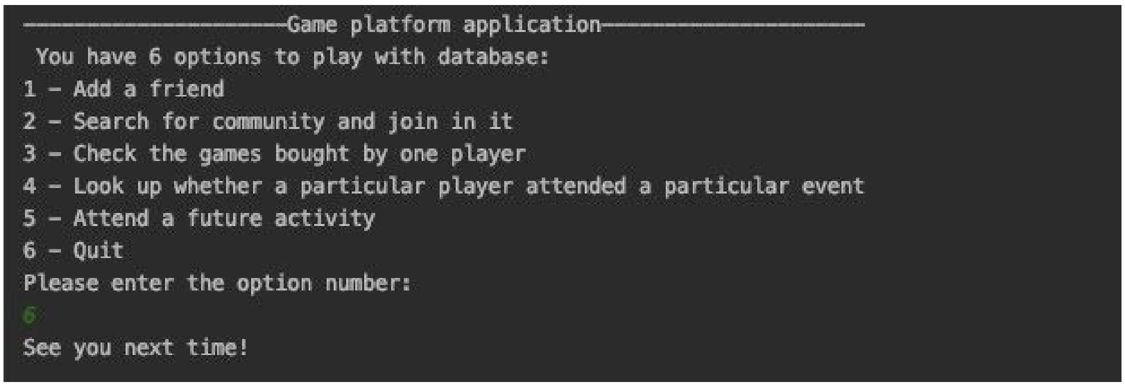

```
---------------------Game platform application---------------------
 You have 6 options to play with database:
1 - Add a friend
2 - Search for community and join in it
3 - Check the games bought by one player
4 - Look up whether a particular player attended a particular event
5 - Attend a future activity
6 - Quit
Please enter the option number:
4
Please enter player id for the search:
14465
Please enter activity name for the search:
Zelda Puzzle Night
We found the following record
Activity name: Zelda Puzzle Night     player_id: 14465     host_time: 2020-09-24 20:00:00
```

5. **Search for future activities and select an activity to attend**
   Part 1: display user the list of future activities
   Part 2: user inputs activity name that he/she wants to attend

   Application demonstration:


```
Hello, welcome to use our awesome application!
Please enter your user id:
15288
---------------------Game platform application---------------------
 You have 6 options to play with database:
1 - Add a friend
2 - Search for community and join in it
3 - Check the games bought by one player
4 - Look up whether a particular player attended a particular event
5 - Attend a future activity
6 - Quit
Please enter the option number:
5
Diamond Hunting - Mario!                                                        2020-07-14 10:00:00.0
Zelda Puzzle Night                                                              2020-09-24 20:00:00.0
Mario game art                                                                  2020-11-01 09:00:00.0
Please enter an activity name you want to join:
Mario game art
You have joined the activity!
```

6. **User quit program.**


```
---------------------Game platform application---------------------
 You have 6 options to play with database:
1 - Add a friend
2 - Search for community and join in it
3 - Check the games bought by one player
4 - Look up whether a particular player attended a particular event
5 - Attend a future activity
6 - Quit
Please enter the option number:
6
See you next time!
```

Question 3:

a. **Search all the posts in a community about one game:**

```
CREATE INDEX post_index ON post (com_name,game_id);
```

By using a composite index named post_index, we can find the posts in a community about one game faster through "SELECT * FROM post WHERE com_name=... AND game_id=..." statement.
We don't use a clustered index because:

1. If we use a clustered index, we need to organize the entire post table based on the indexing key, which is a lot of work.
2. By using composite index we have narrowed down the range of the matching tuples, even if they exist separately in data pages, the cost is relatively small compared with resorting the whole table.

b. **Search games using keywords:**

```
CREATE INDEX gname_index ON game (gname);
CLUSTER game USING gname_index;
```

By creating an index on games' names, a user can quickly search the games they are interested in using keywords in the game store. Because the names of games are not forced to be unique but the publishers will make it characteristic, using clustered indexes can help gather these games with the same keywords together and maximize the cache hits. This is ideal for range queries like "SELECT * FROM game WHERE gname LIKE '%adventure%' ".

Question 4:

CHART 1:

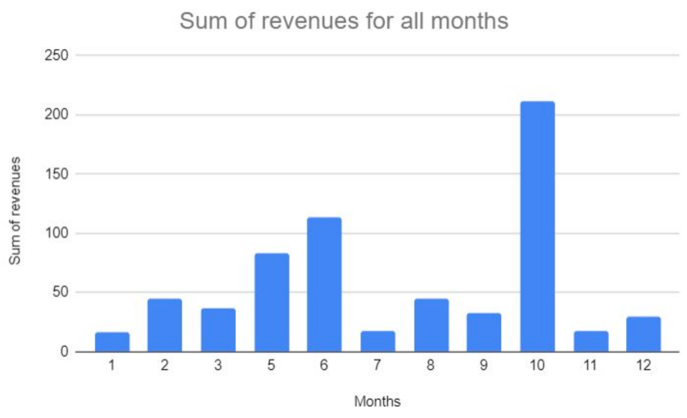
i) SQL

```
SELECT t1.order_id, t1.date. t1.game_id, t2.price FROM
(SELECT order_id, date, game_id FROM buy ) AS t1
JOIN
(SELECT price, game_sid FROM game) AS t2
ON t1.game_id = t2.game_id
)
ORDER BY t1.date ACSE

-- After the resulting queries are exported as csv, we could extract
the month from timestamp with python or excel
-- In some versions it is possible to convert timestamp / datetime
variables to Month type directly, however, we encountered difficulties
```

in casting when using timestamps in pgAdmin. Therefore we decided to move on with python and excel

ii) Chart:



iii) Spreadsheet we worked on:

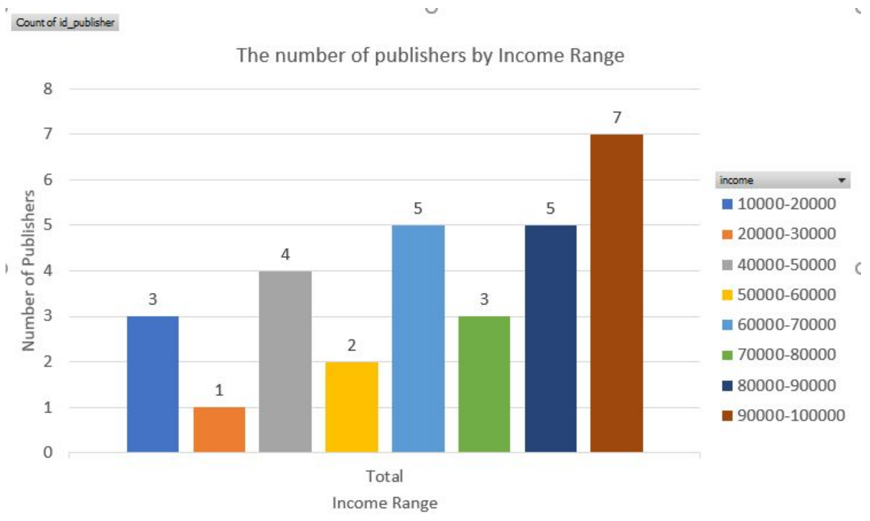
https://docs.google.com/spreadsheets/d/1qYxn6mdrVS7GpU71vAXHxkWUfBfEUnU64VeWt_3NLDQ/edit?usp=sharing

CHART 2:

i) SQL:

```
SELECT * FROM publisher
ORDER BY income ASCE
```

ii) Chart:



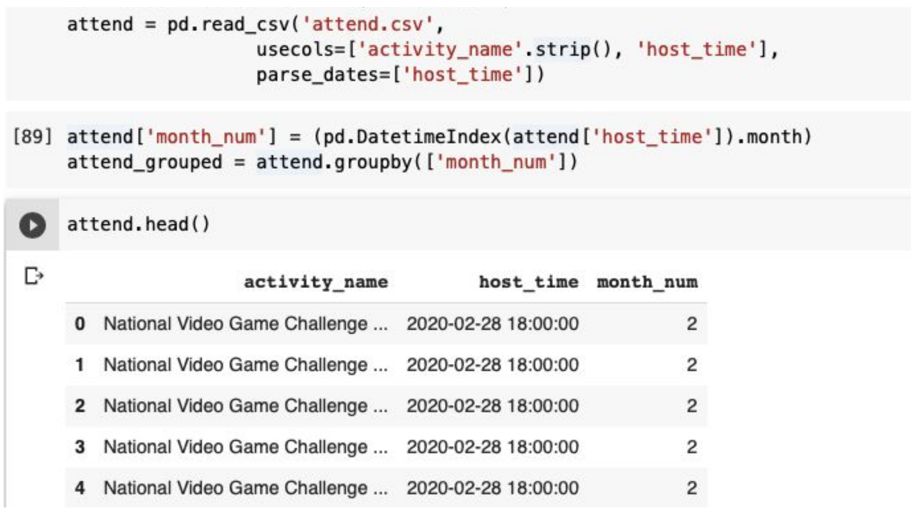
iii) Spreadsheet we worked on (it’s in excel):

https://drive.google.com/file/d/12aLUgzKDicaZMfAzFAsksvwXnfgjRsG-/view?usp=sharing

Question 5: (Creativity)

We are visualizing the data with Python pandas here.
The bar chart is displaying the number of activity attendance per month. This data is processed from the ‘attend’ relation.

This visualization is meaningful as it shows, for example, in which months the activities are more popular (have most attendees).



```
attend = pd.read_csv('attend.csv',
                     usecols=['activity_name'.strip(), 'host_time'],
                     parse_dates=['host_time'])
```

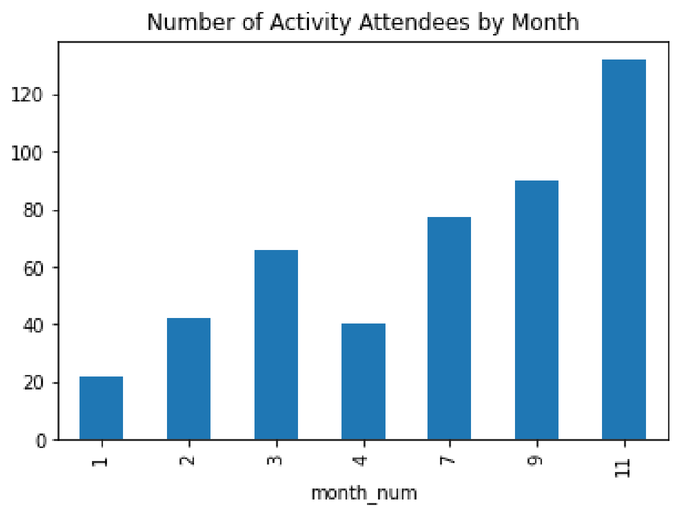
```
[89] attend['month_num'] = (pd.DatetimeIndex(attend['host_time']).month)
     attend_grouped = attend.groupby(['month_num'])
```

```
attend.head()
```

	activity_name	host_time	month_num
0	National Video Game Challenge ...	2020-02-28 18:00:00	2
1	National Video Game Challenge ...	2020-02-28 18:00:00	2
2	National Video Game Challenge ...	2020-02-28 18:00:00	2
3	National Video Game Challenge ...	2020-02-28 18:00:00	2
4	National Video Game Challenge ...	2020-02-28 18:00:00	2

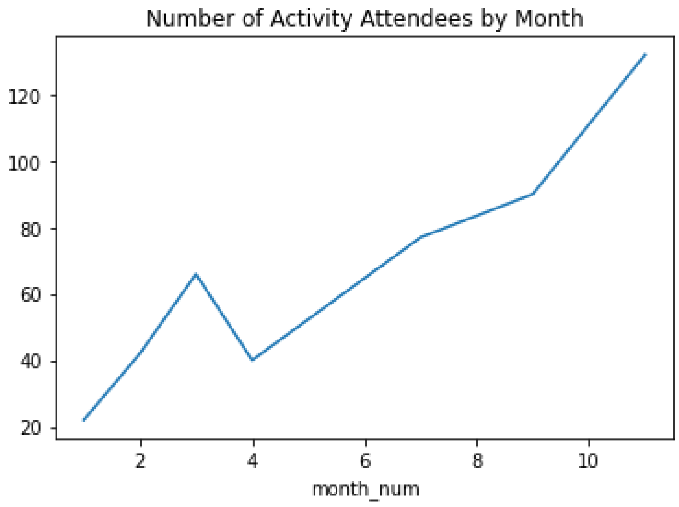
Bar chart:

```
attend_grouped.month_num.sum().plot(kind='bar',x='Month',y='Number of Attendees',title='Number of Activity Attendees by Month')
```



Line chart:

```
attend_grouped.month_num.sum().plot(kind='line',x='Month',y='Number of Attendees',title='Number of Activity Attendees by Month')
```


Number of Activity Attendees by Month
120
100
80
60
40
20
2
4
6
8
10
month_num