Database system laboratory homework 3

**Implementation of Facebook E-R diagram using Python**

2014112123 손준영

2015112241 유지훈

**<Experiment environment>**

**DB Program: MySQL Workbench 8.0.18.0**

**-detail here-**

****

**Host Language: Python 3.7**

|  |
| --- |
| **Library list that I used** |
| Mysql-connector-c 6.1.11  Mysql-connector-python 8.0.18  Mysql-python 1.25 |

Thank you for reading this.

**1. Select four queries from the SQL queries in HW2. Any two queries can’t belong to the same category(I – VI). All the queries should include one or more host variables.**

1) Show the four queries from HW2 you selected.

|  |  |
| --- | --- |
| SQL queries that I selected from HW2 | |
| Query No.1 from HW2 I | Query No.2 from HW2 II |
|  |  |
| Query No.3 from HW IV | |
|  | |
| Query No.4 from HW VI | |
|  | |

2, 3) Write a C(or Python etc) program which can process the four SQL queries. You should include ad many comments as possible to make it easier to be read. And show that your program works.

|  |
| --- |
| Python source of SQL Query No.1 |
|  |
| Results of SQL Query No.1 |
|  |
| Results of Python source of SQL Query No.1 |
|  |

|  |
| --- |
| Python source of SQL Query No.2 |
|  |
| Results of SQL Query No.2 |
|  |
| Results of Python source of SQL Query No.2 |
|  |

|  |
| --- |
| Python source of SQL Query No.3 |
|  |
| Results of SQL Query No.3 |
|  |
| Results of Python source of SQL Query No.3 |
|  |

|  |
| --- |
| Python source of SQL Query No.4 |
|  |
| Results of SQL Query No.4 |
|  |
| Results of Python source of SQL Query No.4 |
|  |

**2. Implement the following.**

1) Define a procedure which contains an aggregate function.

|  |  |
| --- | --- |
| My Procedure SQL source | Description |
| DELIMITER $$  DROP PROCEDURE IF EXISTS get\_like\_count$$  CREATE PROCEDURE get\_like\_count  (  IN in\_user\_id varchar(10)  )  BEGIN  SELECT user\_id, count(liker\_id)  FROM post P natural join likes  GROUP BY user\_id  HAVING user\_id = in\_user\_id;  END$$  DELIMITER ; | This is a procedure that counts the number of ‘Likes’ that other people have expressed in my posts. |
| Result of call this procedure |
| CALL get\_like\_count('0'); |

2) Define a function which returns a value.

|  |  |
| --- | --- |
| My Function SQL source | Description |
| SET GLOBAL log\_bin\_trust\_function\_creators = 1;  DELIMITER $$  DROP FUNCTION IF EXISTS total\_users\_number$$  CREATE FUNCTION total\_users\_number()  RETURNS INT  BEGIN  DECLARE result INT;  SET result = -1;    SELECT count(\*)  INTO result  FROM user;    RETURN result;  END$$  DELIMITER ; | This is a function that returns the total number of accounts which are included at my facebook DB. |
| Result of call this function |
| SELECT total\_users\_number(); |

3,4) Write a C (or Python etc) program which calls/uses both of the procedures/functions defined above. And show that your program works

|  |
| --- |
| Python source of My Procedure |
|  |
| Results of My Procedure at MySQL |
|  |
| Results of Python source of My Procedure |
|  |

|  |
| --- |
| Python source of My Function |
|  |
| Results of SQL Query No.4 |
|  |
| Results of Python source of SQL Query No.4 |
|  |

**3. A program following database.**

|  |
| --- |
| Python source code Facebook class |
| # -\*- coding: utf-8 -\*- import mysql.connector import datetime  #페이스 북 구현 class Facebook:  def \_\_init\_\_(self):  # Connect to my server(localhost)  self.con = mysql.connector.connect(user='root',  password='dgu1234!',  host='localhost',  database='facebook2',  charset='utf8') # utf8 for korean language   # Define cursor variable that will receives results of query  self.cur = self.con.cursor(buffered=True)  self.cur.execute("set names utf8")  self.login\_status = False  self.user\_id = ""  #----------------------------------------------------------------------------------------------------------  def InsertAccount(self, name, birth, phone, gender, address):  # Register Account(Insert   # Checking Logged-in  if (self.login\_status==True):  print ("Please do again After logout")  else:  query = "select user\_id from user;"  self.cur.execute(query)  existing\_user\_id = []   #Give new user ID  for u\_id in self.cur:  existing\_user\_id.append(int(u\_id[0]))  user\_id = max(existing\_user\_id) + 1   query = """insert user values ("{}", "{}", {}, "{}", "{}","{}");""".\  format(str(user\_id), str(name), int(birth), str(phone), str(gender), str(address))  self.cur.execute(query)  self.con.commit()   print("'{}' is registered successfully".format(name))  print("""{}'s user\_id is '{}'""".format(name, user\_id))  # ----------------------------------------------------------------------------------------------------------  def Login(self, user\_id):  #Login to Facebook   # Checking Logged-in  if(self.login\_status == False):  query = "select user\_id from user;"  self.cur.execute(query)  existing\_user\_id = []   # Find existing ID in UserList  for u\_id in self.cur:  existing\_user\_id.append(int(u\_id[0]))  for u\_id in existing\_user\_id:  if(u\_id == int(user\_id)):  self.user\_id = str(user\_id)  self.login\_status = True   if(self.login\_status==True):  print("\nLogin successfully")  else:  print("\nThe user\_id:'{}' is not exist".format(user\_id))   else:  print("\nAlready logged-in. Please login after logout")  # ----------------------------------------------------------------------------------------------------------  def Logout(self):  # Account Logout   # Checking Logged-in  if(self.login\_status==True):  self.user\_id = ""  self.login\_status = False  print("\nLogout successfully")  else:  print("\nAny account is not login ")  # ----------------------------------------------------------------------------------------------------------  def UserInfo(self):  # Print logged-in user's Information   # Checking Logged-in  if(self.login\_status==True):  query = "select \* from user where user\_id='{}';".format(self.user\_id)  self.cur.execute(query)   print("\n-------------------------------AccountInfo-------------------------------")  for (user\_id, name, birth, phone, gender, address) in self.cur:  print("{:2} | {:2} | {:6} | {:14} | {:6} | {:10}".format(  user\_id, name.encode('utf8'), birth, phone.encode('utf8'), gender.encode('utf8'),  address.encode('utf8')))  else:  print("Any account is not login ")  # ----------------------------------------------------------------------------------------------------------  def Posting(self, contents, location="동국대학교"):   # Checking Logged-in  if(self.login\_status==True):  query = "select post\_id from post where user\_id = '{}';".format(self.user\_id)  self.cur.execute(query)  existing\_post\_id = []   #Give new post ID  for p\_id in self.cur:  existing\_post\_id.append(int(p\_id[0]))  if not existing\_post\_id:  post\_id = "00"  else:  post\_id = max(existing\_post\_id)+1  if(post\_id < 10):  post\_id = "0" + str(post\_id)   now = datetime.datetime.now()   # Make a Date Type Consistent  if (now.month < 10):  month = "0"+ str(now.month)  else:  month = str(now.month)  if (now.day < 10):  day = "0" + str(now.day)  else:  day = str(now.day)   date = str(now.year) + str(month) + str(day)   query = """insert post values ("{}","{}",{},"{}","{}")""".\  format(str(post\_id), self.user\_id, int(date), str(location), str(contents))  self.cur.execute(query)  self.con.commit()   print ("\nPosting Successfully")  print ("The post\_id is '{}'".format(post\_id))   else:  print("For posting, you have to login!")  # ----------------------------------------------------------------------------------------------------------  def Comments(self, post\_id, poster\_id, text):  # Comments at post   # Checking login  if (self.login\_status == True):  # Searching post  query = "select count(\*) from post where user\_id ='{}' and post\_id = '{}'".\  format(str(poster\_id), str(post\_id))  self.cur.execute(query)  for n in self.cur:  num = int(n[0])  if (num == 0):  print("The post is not exist.")  # If Post exists  else:  query = "select comment\_id from comments where user\_id ='{}' and post\_id = '{}'".\  format(str(poster\_id), str(post\_id))  self.cur.execute(query)   # Give a new comment\_id  existing\_comment\_id = []  for comment\_id in self.cur:  existing\_comment\_id.append(int(comment\_id[0]))  if not existing\_comment\_id:  comment\_id = "000"  else:  comment\_id = max(existing\_comment\_id) + 1  if (comment\_id < 10):  comment\_id = "00" + str(comment\_id)  elif (10 <= comment\_id < 100):  comment\_id = "0" + str(comment\_id)   now = datetime.datetime.now()   # Make a Date Type Consistent  if (now.month < 10):  month = "0" + str(now.month)  else:  month = str(now.month)  if (now.day < 10):  day = "0" + str(now.day)  else:  day = str(now.day)   date = str(now.year) + str(month) + str(day)   query = "select (select name from user where user\_id = '{}') as name, post\_id, contents from post where user\_id ='{}' and post\_id = '{}'"\  .format(str(poster\_id), str(poster\_id), str(post\_id))  self.cur.execute(query)   for name, post\_id, contents in self.cur:  print("\n\nname: '{:1}' | post\_id : '{:1}' \n '{:5}'".  format(name.encode('utf8'), post\_id, contents.encode('utf8')))    query = "insert comments values ('{}','{}','{}','{}', {}, '{}')".\  format(str(comment\_id), str(post\_id), str(poster\_id), str(self.user\_id), int(date), str(text))  self.cur.execute(query)  self.con.commit()  print ("Commenting on the post successfully")  print ("-------------------------------Comments-------------------------------")  query = "select comment\_id, text from comments where user\_id = '{}' and post\_id='{}'".\  format(poster\_id, post\_id)  self.cur.execute(query)  for comment\_id, text in self.cur:  print("'{:2}' | '{:6}'".format(comment\_id, text.encode('utf8')))     else:  print("\nFor commenting on the post, you have to login")  # ----------------------------------------------------------------------------------------------------------  def Like(self, post\_id, poster\_id):  if (self.login\_status == True):  # Searching post  query = "select count(\*) from post where user\_id ='{}' and post\_id = '{}'".format(str(poster\_id),  str(post\_id))  self.cur.execute(query)  for n in self.cur:  num = int(n[0])  if (num == 0):  print("The post is not exist.")  # If Post exists  else:  query = "select like\_id from likes where user\_id ='{}' and post\_id = '{}'".format(str(poster\_id),  str(post\_id))  self.cur.execute(query)   # Give a new comment\_id  existing\_like\_id = []  for like\_id in self.cur:  existing\_like\_id.append(int(like\_id[0]))  if not existing\_like\_id:  like\_id = "0000"  else:  like\_id = max(existing\_like\_id) + 1  if (like\_id < 10):  like\_id = "000" + str(like\_id)  elif (10 <= like\_id < 100):  like\_id = "00" + str(like\_id)  elif (100 <= like\_id < 1000):  like\_id = "0" + str(like\_id)   query = "select (select name from user where user\_id = '{}') as name, post\_id, contents from post where user\_id ='{}' and post\_id = '{}'"\  .format(str(poster\_id), str(poster\_id), str(post\_id))  self.cur.execute(query)   for name, post\_id, contents in self.cur:  print("\n\nname: '{:1}' | post\_id : '{:1}' \n '{:5}'".format(name.encode('utf8'), post\_id,  contents.encode('utf8')))   query = "select liker\_id, user\_id, post\_id from likes where user\_id ='{}' and post\_id='{}'"\  .format(poster\_id, post\_id)  self.cur.execute(query)   # For checking like already  check = 0  for liker\_id, \_, \_ in self.cur:  if (liker\_id == self.user\_id):  print("Already you did Like")  check = 1  query = "select count(\*) from likes where user\_id='{}' and post\_id='{}'"\  .format(poster\_id, post\_id)  self.cur.execute(query)  for num\_likes in self.cur:  print("Like:{}".format(int(num\_likes[0])))  break   if(check == 0):  query = "insert likes values ('{}', '{}', '{}', '{}')".format(like\_id, post\_id, poster\_id, self.user\_id)  self.cur.execute(query)  self.con.commit()  print ("Like on the post successfully")   query = "select count(\*) from likes where user\_id='{}' and post\_id='{}'".format(poster\_id, post\_id)  self.cur.execute(query)  for num\_likes in self.cur:  print("Like:{}".format(int(num\_likes[0])))   else:  print("\nFor like on the post, you have to login")  # ----------------------------------------------------------------------------------------------------------  def Timeline(self):  if (self.login\_status == True):  print("\n---------------------------Timeline---------------------------")  query = "select name, user\_id, post\_id, date, contents from post natural join user order by date desc"  self.cur.execute(query)  for name, user\_id, post\_id, date, contents in self.cur:  print("name: '{}' | poster\_id: '{}' | post\_id: '{}' | date: '{}' \n contents: '{}'").\  format(name.encode('utf8'), user\_id, post\_id, date, contents.encode('utf8'))  else:  print("For showing, you have to login")  #----------------------------------------------------------------------------------------------------------  facebook = Facebook() |

|  |
| --- |
| 1. Facebook page of Register Account |
|  |
| Method “InsertAccount()” |
| def InsertAccount(self, name, birth, phone, gender, address):  # Register Account(Insert   # Checking Logged-in  if (self.login\_status==True):  print ("Please do again After logout")  else:  query = "select user\_id from user;"  self.cur.execute(query)  existing\_user\_id = []   #Give new user ID  for u\_id in self.cur:  existing\_user\_id.append(int(u\_id[0]))  user\_id = max(existing\_user\_id) + 1   query = """insert user values ("{}", "{}", {}, "{}", "{}","{}");""".\  format(str(user\_id), str(name), int(birth), str(phone), str(gender), str(address))  self.cur.execute(query)  self.con.commit()   print("'{}' is registered successfully".format(name))  print("""{}'s user\_id is '{}'""".format(name, user\_id)) |
| facebook.InsertAccount("김지민","19950602","01000001234","남자","중앙동") facebook.InsertAccount("김가영","19970604","01022221234","여자","필동") |
| Results of Python method “InsertAccount()” |
|  |
| Before execution |
|  |
| After execution |
|  |
| 2. Facebook page of login |
|  |
| Method “Login()” |
| def Login(self, user\_id):  #Login to Facebook   # Checking Logged-in  if(self.login\_status == False):  query = "select user\_id from user;"  self.cur.execute(query)  existing\_user\_id = []   # Find existing ID in UserList  for u\_id in self.cur:  existing\_user\_id.append(int(u\_id[0]))  for u\_id in existing\_user\_id:  if(u\_id == int(user\_id)):  self.user\_id = str(user\_id)  self.login\_status = True   if(self.login\_status==True):  print("\nLogin successfully")  else:  print("\nThe user\_id:'{}' is not exist".format(user\_id))   else:  print("\nAlready logged-in. Please login after logout") |
| Facebook.Login(2)  facebook.Login(3) |
| Results of Python method “Login()” |
|  |
| Results When already login |
| Facebook.Login(2)  facebook.Login(3) |
|  |
| 3. Facebook page of Logout |
|  |
| Method “Logout()” |
| def Logout(self):  # Account Logout   # Checking Logged-in  if(self.login\_status==True):  self.user\_id = ""  self.login\_status = False  print("\nLogout successfully")  else:  print("\nAny account is not login ")  facebook.Login(1) facebook.Logout() |
| Results of Python method “Logout()” |
|  |

|  |
| --- |
| 4. Facebook page of User Information |
|  |
| Method “UserInfo()” |
| def UserInfo(self):  # Print logged-in user's Information   # Checking Logged-in  if(self.login\_status==True):  query = "select \* from user where user\_id='{}';".format(self.user\_id)  self.cur.execute(query)   print("\n-------------------------------AccountInfo-------------------------------")  for (user\_id, name, birth, phone, gender, address) in self.cur:  print("{:2} | {:2} | {:6} | {:14} | {:6} | {:10}".format(  user\_id, name.encode('utf8'), birth, phone.encode('utf8'), gender.encode('utf8'),  address.encode('utf8')))  else:  print("Any account is not login ") |
| facebook.Login(1) facebook.UserInfo() |
| Results of Python method “UserInfo()” |
|  |

|  |
| --- |
| 5. Facebook page of Posting |
|  |
| Method “Posting()” |
| def Posting(self, contents, location="동국대학교"):   # Checking Logged-in  if(self.login\_status==True):  query = "select post\_id from post where user\_id = '{}';".format(self.user\_id)  self.cur.execute(query)  existing\_post\_id = []   #Give new post ID  for p\_id in self.cur:  existing\_post\_id.append(int(p\_id[0]))  if not existing\_post\_id:  post\_id = "00"  else:  post\_id = max(existing\_post\_id)+1  if(post\_id < 10):  post\_id = "0" + str(post\_id)   now = datetime.datetime.now()   # Make a Date Type Consistent  if (now.month < 10):  month = "0"+ str(now.month)  else:  month = str(now.month)  if (now.day < 10):  day = "0" + str(now.day)  else:  day = str(now.day)   date = str(now.year) + str(month) + str(day)   query = """insert post values ("{}","{}",{},"{}","{}")""".\  format(str(post\_id), self.user\_id, int(date), str(location), str(contents))  self.cur.execute(query)  self.con.commit()   print ("\nPosting Successfully")  print ("The post\_id is '{}'".format(post\_id))   else:  print("For posting, you have to login!") |
| facebook.Login(1) facebook.Posting("데이터베이스 과제하는중!!!","신공학관") |
| Results of Python method “Posting()” |
|  |
| Before execution |
|  |
| After execution |
|  |

|  |
| --- |
| 6. Facebook page of Comments |
|  |
| Method “Comments()” |
| def Comments(self, post\_id, poster\_id, text):  # Comments at post   # Checking login  if (self.login\_status == True):  # Searching post  query = "select count(\*) from post where user\_id ='{}' and post\_id = '{}'".\  format(str(poster\_id), str(post\_id))  self.cur.execute(query)  for n in self.cur:  num = int(n[0])  if (num == 0):  print("The post is not exist.")  # If Post exists  else:  query = "select comment\_id from comments where user\_id ='{}' and post\_id = '{}'".\  format(str(poster\_id), str(post\_id))  self.cur.execute(query)   # Give a new comment\_id  existing\_comment\_id = []  for comment\_id in self.cur:  existing\_comment\_id.append(int(comment\_id[0]))  if not existing\_comment\_id:  comment\_id = "000"  else:  comment\_id = max(existing\_comment\_id) + 1  if (comment\_id < 10):  comment\_id = "00" + str(comment\_id)  elif (10 <= comment\_id < 100):  comment\_id = "0" + str(comment\_id)   now = datetime.datetime.now()   # Make a Date Type Consistent  if (now.month < 10):  month = "0" + str(now.month)  else:  month = str(now.month)  if (now.day < 10):  day = "0" + str(now.day)  else:  day = str(now.day)   date = str(now.year) + str(month) + str(day)  query = "select (select name from user where user\_id = '{}') as name, post\_id, contents from post where user\_id ='{}' and post\_id = '{}'"\  .format(str(poster\_id), str(poster\_id), str(post\_id))  self.cur.execute(query)   for name, post\_id, contents in self.cur:  print("\n\nname: '{:1}' | post\_id : '{:1}' \n '{:5}'".  format(name.encode('utf8'), post\_id, contents.encode('utf8')))    query = "insert comments values ('{}','{}','{}','{}', {}, '{}')".\  format(str(comment\_id), str(post\_id), str(poster\_id), str(self.user\_id), int(date), str(text))  self.cur.execute(query)  self.con.commit()  print ("Commenting on the post successfully")  print ("-------------------------------Comments-------------------------------")  query = "select comment\_id, text from comments where user\_id = '{}' and post\_id='{}'".\  format(poster\_id, post\_id)  self.cur.execute(query)  for comment\_id, text in self.cur:  print("'{:2}' | '{:6}'".format(comment\_id, text.encode('utf8')))     else:  print("\nFor commenting on the post, you have to login") |
| facebook.Comments("02","1", "ㅠㅠ힘들겠다 조금만힘내자!")  facebook.Comments("02","1", "얼른 과제끝내고 맛있는거 먹자 ㅎㅅㅎ") |
| Results of Python method “Comments()” |
|  |
| Before execution |
|  |
| After execution |
|  |

|  |
| --- |
| 7. Facebook page of Like |
|  |
| Method “Like()” |
| def Like(self, post\_id, poster\_id):  if (self.login\_status == True):  # Searching post  query = "select count(\*) from post where user\_id ='{}' and post\_id = '{}'".format(str(poster\_id),  str(post\_id))  self.cur.execute(query)  for n in self.cur:  num = int(n[0])  if (num == 0):  print("The post is not exist.")  # If Post exists  else:  query = "select like\_id from likes where user\_id ='{}' and post\_id = '{}'".format(str(poster\_id),  str(post\_id))  self.cur.execute(query)   # Give a new comment\_id  existing\_like\_id = []  for like\_id in self.cur:  existing\_like\_id.append(int(like\_id[0]))  if not existing\_like\_id:  like\_id = "0000"  else:  like\_id = max(existing\_like\_id) + 1  if (like\_id < 10):  like\_id = "000" + str(like\_id)  elif (10 <= like\_id < 100):  like\_id = "00" + str(like\_id)  elif (100 <= like\_id < 1000):  like\_id = "0" + str(like\_id)   query = "select (select name from user where user\_id = '{}') as name, post\_id, contents from post where user\_id ='{}' and post\_id = '{}'"\  .format(str(poster\_id), str(poster\_id), str(post\_id))  self.cur.execute(query)   for name, post\_id, contents in self.cur:  print("\n\nname: '{:1}' | post\_id : '{:1}' \n '{:5}'".format(name.encode('utf8'), post\_id,  contents.encode('utf8')))   query = "select liker\_id, user\_id, post\_id from likes where user\_id ='{}' and post\_id='{}'"\  .format(poster\_id, post\_id)  self.cur.execute(query)   # For checking like already  check = 0  for liker\_id, \_, \_ in self.cur:  if (liker\_id == self.user\_id):  print("Already you did Like")  check = 1  query = "select count(\*) from likes where user\_id='{}' and post\_id='{}'"\  .format(poster\_id, post\_id)  self.cur.execute(query)  for num\_likes in self.cur:  print("Like:{}".format(int(num\_likes[0])))  break   if(check == 0):  query = "insert likes values ('{}', '{}', '{}', '{}')".format(like\_id, post\_id, poster\_id, self.user\_id)  self.cur.execute(query)  self.con.commit()  print ("Like on the post successfully")   query = "select count(\*) from likes where user\_id='{}' and post\_id='{}'".format(poster\_id, post\_id)  self.cur.execute(query)  for num\_likes in self.cur:  print("Like:{}".format(int(num\_likes[0])))   else:  print("\nFor like on the post, you have to login") |
| facebook.Login(2)  facebook.Like("02","1")  #If One more check like facebook.Like("02","1") |
| Results of Python method “Like()” |
|  |
| Before execution |
|  |
| After execution |
|  |

|  |
| --- |
| 8. Facebook page of Timeline |
|  |
| Method “Timeline()” |
| def Timeline(self):  if (self.login\_status == True):  print("\n---------------------------Timeline---------------------------")  query = "select name, user\_id, post\_id, date, contents from post natural join user order by date desc"  self.cur.execute(query)  for name, user\_id, post\_id, date, contents in self.cur:  print("name: '{}' | poster\_id: '{}' | post\_id: '{}' | date: '{}' \n contents: '{}'").\  format(name.encode('utf8'), user\_id, post\_id, date, contents.encode('utf8'))  else:  print("For showing, you have to login") |
| facebook.Login(2)  facebook.Timeline() |
| Results of Python method “Timeline()” |
|  |