

DBS : Term Projectet

(중고차 판매 시스템)



충북대학교
CHUNGBUK NATIONAL UNIVERSITY

2019038003

고영민

목 차

1. 요구사항 정의서

1-1. Entity	3
1-2. Relation	4

2. ER Diagram

2-1. ER Diagram	5
-----------------------	---

3. Relation Schema

3-1. Relation Schema	6
----------------------------	---

4. Relation Schema Normalization

4-1. Relation Schema Normalization	6
--	---

5. DataBase 설명

5-1. User Table	7
5-2. Staff Table	8
5-3. Car Table	9
5-4. Accident Table	10

6. 프로그램 설명

6-1. MainWindow	11
6-2. Registration Window	12
6-3. UserWindow	13
6-4. StaffWindow	17

Project GitHub URL : https://github.com/Go-YM/DBS_TermProject.git

1. 요구사항 정의서

Entity

- User

1. User는 uid(User 아이디), pw(비밀번호), email(이메일), user_name(이름), license(운전면허증 유무)의 정보를 갖는다.
2. User의 Primary key는 uid이다.
3. User의 uid, email, user_name의 자료형은 문자열이다.
4. User의 pw의 자료형은 정수이고, 4자리수로 구성되어 있다.

- Staff

1. Staff는 sid(Staff 아이디), pw(비밀번호), email(이메일), user_name(이름)의 정보를 갖는다.
2. Staff의 Primary key는 sid이다.
3. Staff의 sid, email, user_name의 자료형은 문자열이다.
4. Staff의 pw의 자료형은 정수이고, 4자리수로 구성되어 있다.

- Car

1. Car는 cid(Car 아이디), age(연식), model(차종), distance(주행거리), price(가격), image(사진)의 정보를 갖는다.
2. Car의 Primary key는 cid이다.
3. Car의 cid, model, seller, buyer, image의 자료형은 문자열이다.
4. Car의 age, distance, price의 자료형은 정수이다.
5. Car의 image는 사진의 경로 정보를 갖는다.
6. Car는 buyer(User), seller(Staff)라는 foreign key를 갖는다.

- Accident

1. Accident는 anum(사고 일련번호), adate(사고 일자)의 정보를 갖는다.
2. Accident의 Partial key는 anum이다.
3. Accident의 anum의 자료형은 정수이다.
4. Accident의 adate의 자료형은 문자열이다.
5. Accident는 Car를 상위 Entity로 갖는 Weak Entity이다.

Relation

- Book

1. User와 Car는 N:M 관계를 갖는다.
2. User는 Car에 대한 정보들을 열람할 수 있다.
3. User는 Car를 보고 마음에 드는 Car를 예약할 수 있다.

- Register

1. Staff와 Car는 N:M 관계를 갖는다.
2. Staff는 새로운 Car에 대한 정보를 등록할 수 있다.

- Delete

1. Staff와 Car는 N:M 관계를 갖는다.
2. Staff는 기존의 Car에 대한 정보를 삭제할 수 있다.

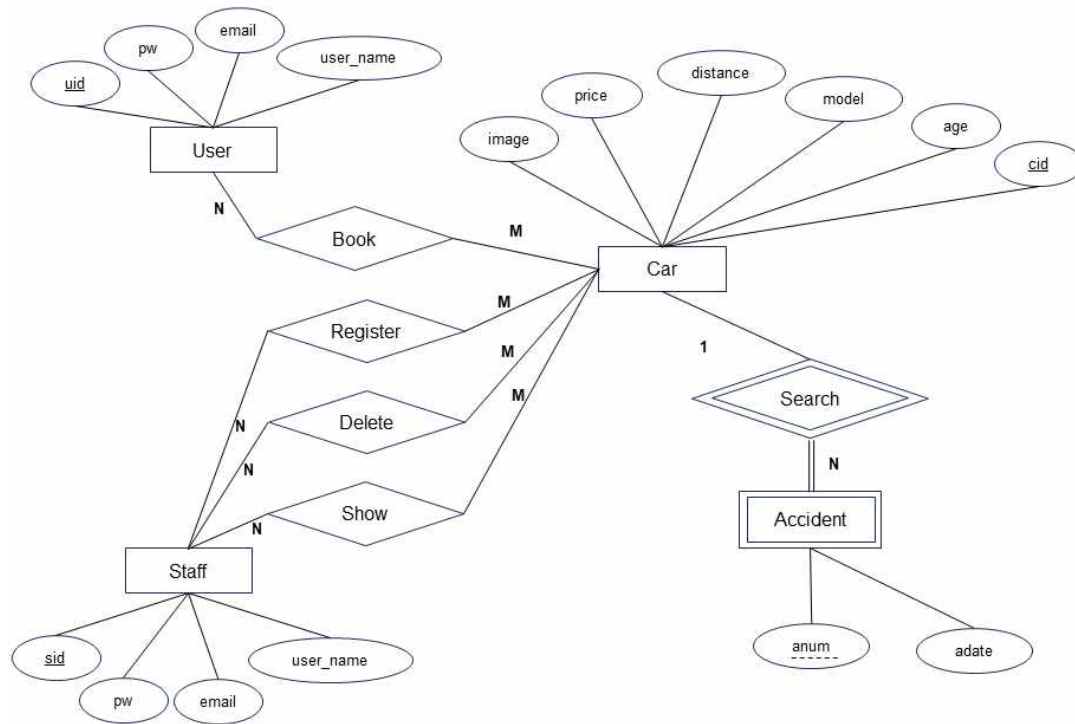
- Show

1. Staff와 Car는 N:M 관계를 갖는다.
2. Staff는 User가 예약한 Car에 대한 정보와 User에 대한 정보를 열람 할 수 있다.

- Search

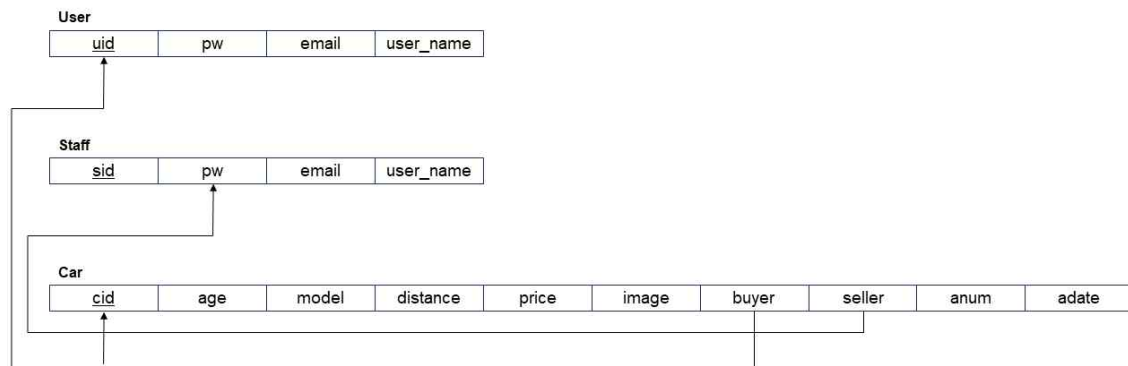
1. Car와 Accident는 1:N 관계를 갖는다.
2. Accident는 Car의 Weak Entity로 Weak Relation으로 표기된다.
3. Car에 대한 Accident를 조회할 수 있다.

2. ER Diagram



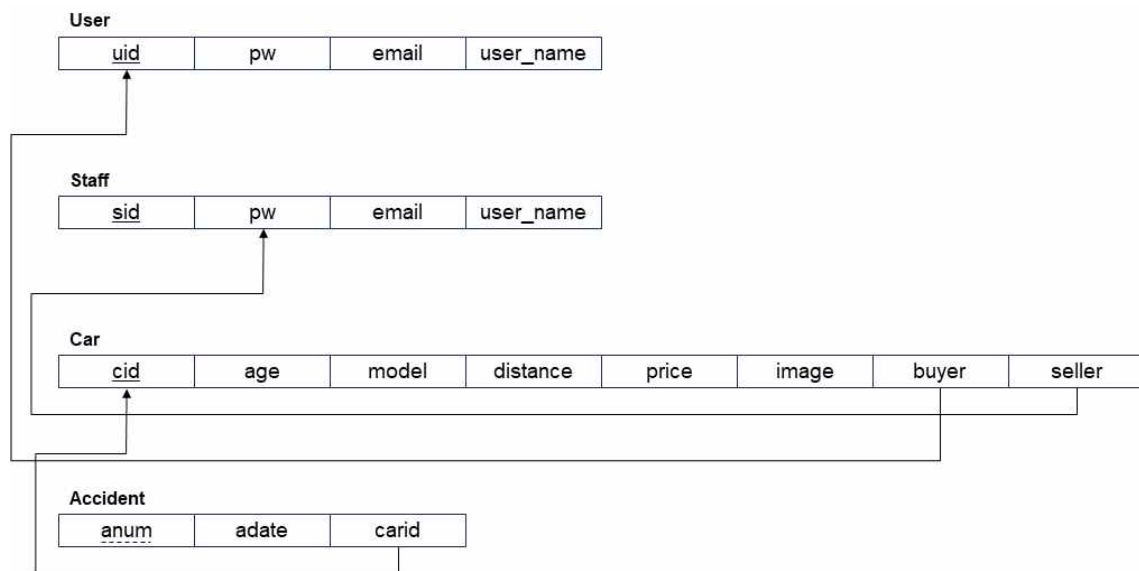
- Book Relation은 N:M 관계를 갖고, User의 uid를 Car가 foreign key로 받는다.
- Register, Delete, Show Relation은 N:M 관계를 갖고, Staff의 sid를 Car가 foreign key로 받는다.
- Search Relation은 1:N의 관계를 갖는 Weak Relation이다. 이는 Car와 Weak Entity인 Accident의 관계이기에 그렇다. Accident는 Car에 종속되어있기에 Car의 Primary Key인 cid를 받아오고, Partial Key인 anum을 사용한다.

3. Relation Schema



- Car Table에서의 attribute 중 anum (사고 일련번호), adate(사고 날짜)를 새로운 Table로 분리하였다.

4. Relation Schema Normalization (3NF)



5. Database 설명

- User Table

<Table 구조>

```
mysql> DESCRIBE User;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| uid        | varchar(255)  | NO   | PRI | NULL    |       |
| pw         | int(4)        | YES  |     | NULL    |       |
| email      | varchar(255)  | YES  |     | NULL    |       |
| user_name  | varchar(255)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

<Table 출력>

```
mysql> SELECT * FROM User;
+-----+-----+-----+-----+-----+
| uid  | pw  | email          | user_name | license |
+-----+-----+-----+-----+-----+
| user1 | 1234 | user1@gmail.com | user1     | NULL    |
| user2 | 1234 | user2@gmail.com | user2     | NULL    |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

- Staff Table

<Table 구조>

```
mysql> DESCRIBE Staff;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| sid        | varchar(255)  | NO   | PRI | NULL     |       |
| pw         | int(4)        | YES  |     | NULL     |       |
| email      | varchar(255)  | YES  |     | NULL     |       |
| user_name  | varchar(255)  | YES  |     | NULL     |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

<Table 출력>

```
mysql> SELECT * FROM Staff;
+-----+-----+-----+-----+
| sid    | pw    | email                | user_name |
+-----+-----+-----+-----+
| cdj0223 | 5678 | choidg@gmail.com    | 최 동 진  |
| gym0117 | 1234 | goym@gmail.com      | 고 영 민  |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```


- Car Table

<Table 구조>

```
mysql> DESCRIBE Car;
```

Field	Type	Null	Key	Default	Extra
cid	varchar(255)	NO	PRI	NULL	
age	int(11)	YES		NULL	
model	varchar(255)	YES		NULL	
seller	varchar(255)	YES	MUL	NULL	
distance	int(11)	YES		NULL	
price	int(11)	YES		NULL	
image	varchar(255)	YES		NULL	
buyer	varchar(255)	YES	MUL	NULL	

8 rows in set (0.00 sec)

<Table 출력>

```
mysql> SELECT * FROM Car;
```

cid	age	model	seller	distance	price
01라 3164	2019	벨로스터	gym0117	25000	1180
01라 3164.jpg		user1			
01주 5031	2014	K3	cdj0223	158444	450
01주 5031.jpg		user1			
02구 1037	2019	더 뉴 아반떼	gym0117	40745	1450
02구 1037.jpg		user1			
04투 8107	2019	아이오닉 일렉트릭 Q	gym0117	36862	1950
04투 8107.jpg		NULL			
08모 0717	2016	아반떼 AD	gym0117	75238	1149
50노 8360	2016	디스커버리	cdj0223	94790	3170
50노 8360.jpg		NULL			
52버 2771	2018	아반떼 AD	gym0117	61657	1280
52버 2771.jpg		NULL			
53오 1566	2017	bmw 520d	cdj0223	131408	2750
53오 1566.jpg		NULL			
54나 1831	2018	더 뉴 K5	cdj0223	85544	1299
54나 1831.jpg		NULL			
59고 0416	2016	아반떼 AD	gym0117	72705	1230
59고 0416.jpg		NULL			
61우 6715	2008	아반떼 HD	cdj0223	77625	460
61우 6715.jpg		NULL			
64조 8889	2011	아반떼 MD	gym0117	157104	590
64조 8889.jpg		NULL			
65투 0989	2015	그랜저 HG	gym0117	70934	1550
65투 0989.jpg		NULL			

50 rows in set (0.00 sec)

- Accident Table

<Table 구조>

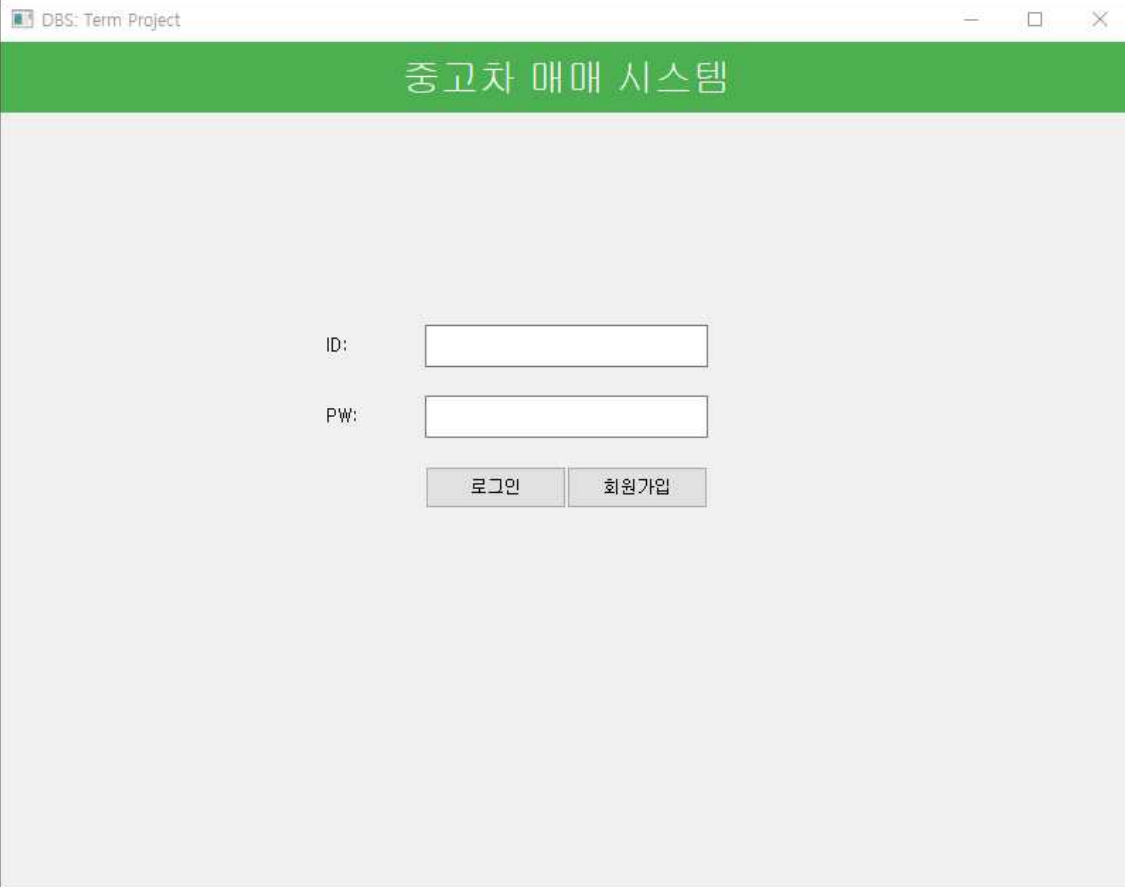
```
mysql> DESCRIBE Accident;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| anum  | int(11)       | NO   | PRI | NULL    |       |
| adate | varchar(255)  | YES  |     | NULL    |       |
| car_id | varchar(255)  | YES  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

<Table 출력>

```
mysql> SELECT * FROM Accident;
+-----+-----+-----+
| anum | adate      | car_id |
+-----+-----+-----+
| 3132 | 2021-03-21 | 10# 6455 |
| 6474 | 2022-06-08 | 01# 5031 |
| 8320 | 2022-07-06 | 33# 8889 |
| 8675 | 2022-09-25 | 10# 6455 |
| 9134 | 2022-10-27 | 65# 0989 |
| 10432 | 2023-01-01 | 52# 2771 |
| 11798 | 2023-03-04 | 24# 1853 |
| 16532 | 2023-12-05 | 24# 5776 |
+-----+-----+-----+
8 rows in set (0.00 sec)
```

6. 프로그램 설명 (Using Python)

1. MainWindow



```
def check_login_User(id, pw):
    try:
        remote = mysql.connector.connect(
            host="192.168.56.101",
            user="goym",
            password="your_password",
            database="TermDB",
            port=4567
        )

        cur = remote.cursor()

        query = f"SELECT * FROM User WHERE uid = '{id}' AND pw = '{pw}'"
        cur.execute(query)
        result = cur.fetchall()

        remote.close()

        return len(result) > 0

    except Exception as e:
        print(f"Error: {e}")
        return False

def check_login_Staff(id, pw):
    try:
        remote = mysql.connector.connect(
            host="192.168.56.101",
            user="goym",
            password="your_password",
            database="TermDB",
            port=4567
        )

        cur = remote.cursor()

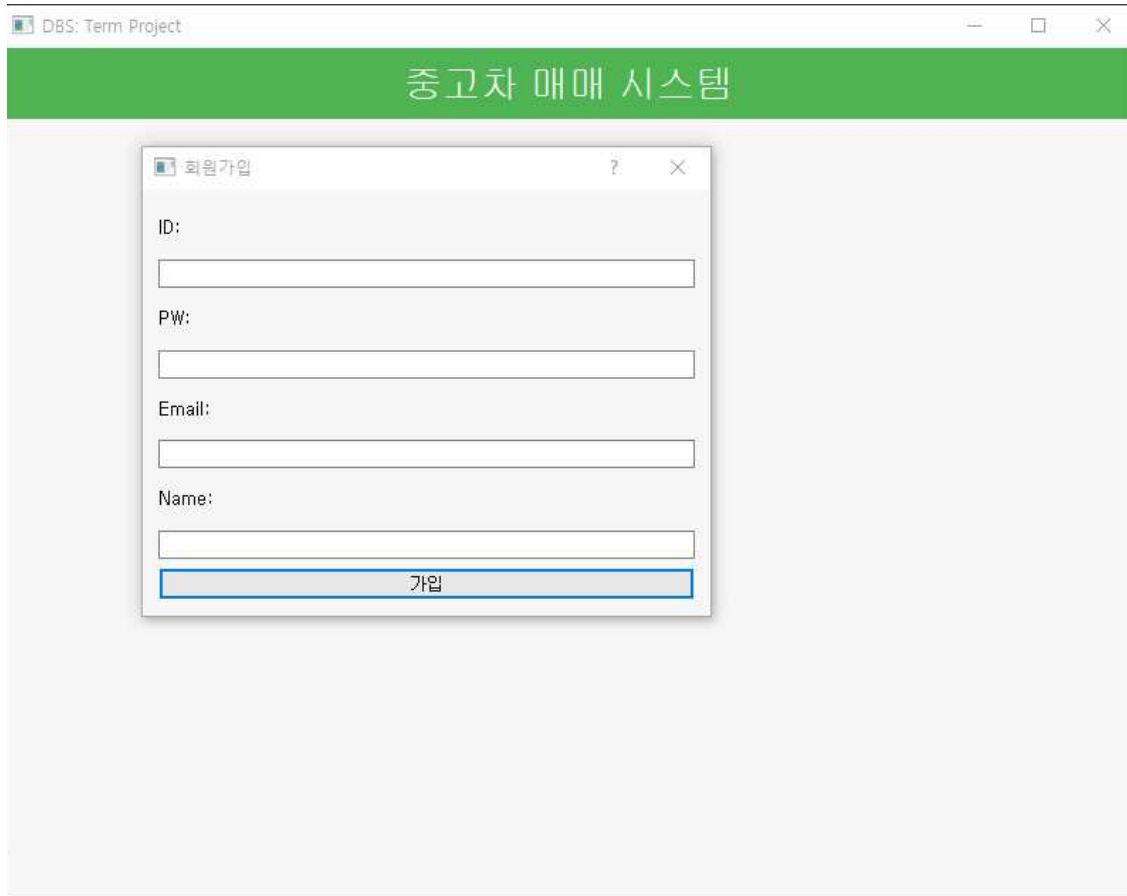
        query = f"SELECT * FROM Staff WHERE sid = '{id}' AND pw = '{pw}'"
        cur.execute(query)
        result = cur.fetchall()

        remote.close()

        return len(result) > 0

    except Exception as e:
        print(f"Error: {e}")
        return False
```

2. RegistrationWindow



```
def register_new_user(id, pw, email, name):
    try:
        remote = mysql.connector.connect(
            host="192.168.56.101",
            user="goym",
            password="your_password",
            database="TermDB",
            port=4567
        )

        cur = remote.cursor()

        check_query = f"SELECT uid FROM User WHERE uid = '{id}'"
        cur.execute(check_query)
        existing_user = cur.fetchone()

        if existing_user:
            return False
        else:
            insert_query = f"INSERT INTO User (uid, pw, email, user_name) VALUES ('{id}', '{pw}', '{email}', '{name}')"
            cur.execute(insert_query)
            remote.commit()
            return True

    except Exception as e:
        print(f"Error: {e}")
        return False

    finally:
        cur.close()
        remote.close()
```

3. User Window

DBS: Term Project - User Window

중고차 매매 시스템 - User



차량 : 벨로스터

판매자 : 고영민 딜러

가격 : 1180 만원

2019년식 25000km

예약하기

조회하기

```
def get_car_data():
    try:
        remote = mysql.connector.connect(
            host="192.168.56.101",
            user="goym",
            password="your_password",
            database="TermDB",
            port=4567
        )

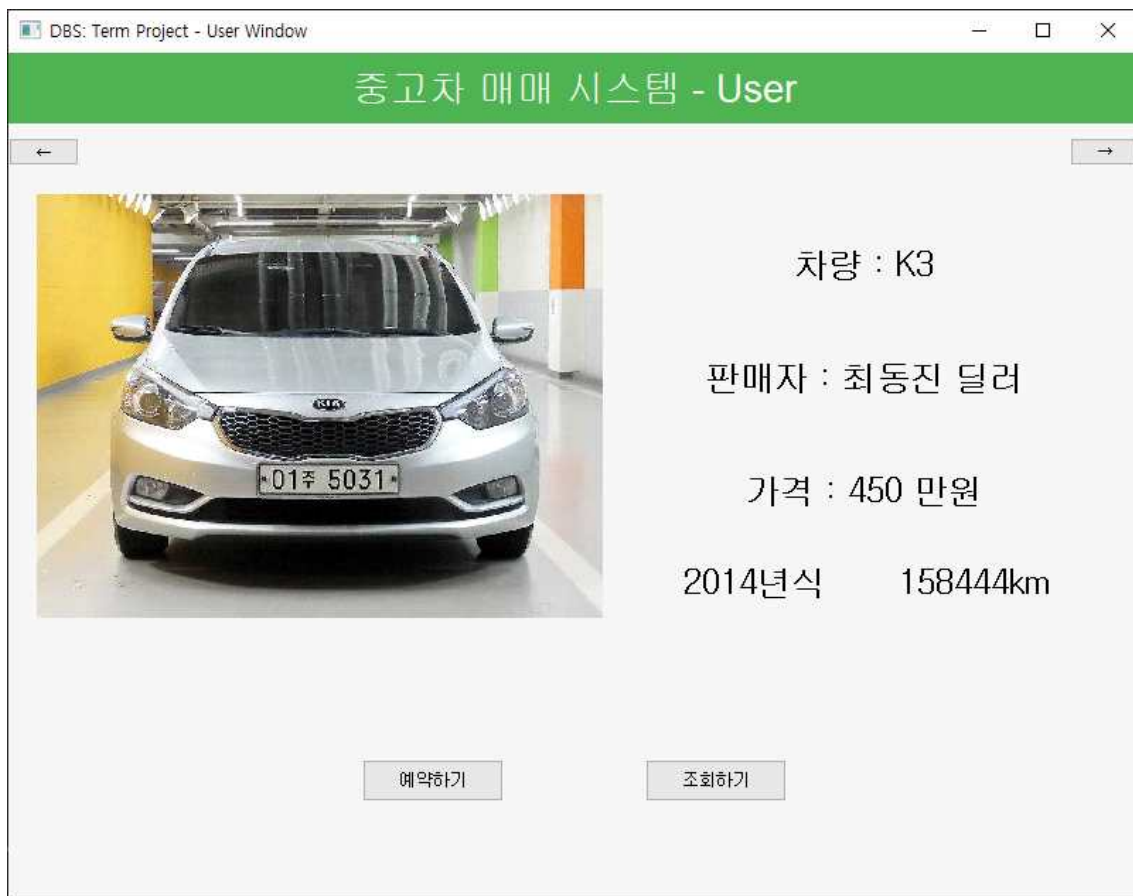
        cur = remote.cursor(dictionary=True) # Use dictionary cursor

        # Car 테이블의 모든 데이터를 가져옴
        cur.execute("SELECT * FROM Car")
        car_data = cur.fetchall()

        return car_data

    except Exception as e:
        print(f"Error: {e}")
        return None

    finally:
        cur.close()
        remote.close()
```



```
def get_car_data():
    try:
        remote = mysql.connector.connect(
            host="192.168.56.101",
            user="goym",
            password="your_password",
            database="TermDB",
            port=4567
        )

        cur = remote.cursor(dictionary=True) # Use dictionary cursor

        # Car 테이블의 모든 데이터를 가져옴
        cur.execute("SELECT * FROM Car")
        car_data = cur.fetchall()

        return car_data

    except Exception as e:
        print(f"Error: {e}")
        return None

    finally:
        cur.close()
        remote.close()
```

중고차 매매 시스템 - User

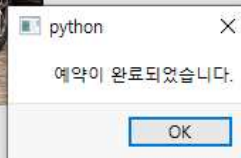


차량 : 미니 쿠퍼

판매자 : 최동진 딜러

가격 : 2100 만원

19721km



예약하기

조회하기

```
def reserve_car(cid, uid):
    try:
        remote = mysql.connector.connect(
            host="192.168.56.101",
            user="goym",
            password="your_password",
            database="TermDB",
            port=4567
        )

        cur = remote.cursor(dictionary=True)

        cur.execute("UPDATE Car SET buyer = %s WHERE cid = %s", (uid, cid))
        remote.commit()

        return True

    except Exception as e:
        print(f"Error: {e}")
        return False

    finally:
        cur.close()
        remote.close()
```




```
def check_accident(cid):
    try:
        remote = mysql.connector.connect(
            host="192.168.56.101",
            user="goym",
            password="your_password",
            database="TermDB",
            port=4567
        )

        cur = remote.cursor(dictionary=True)

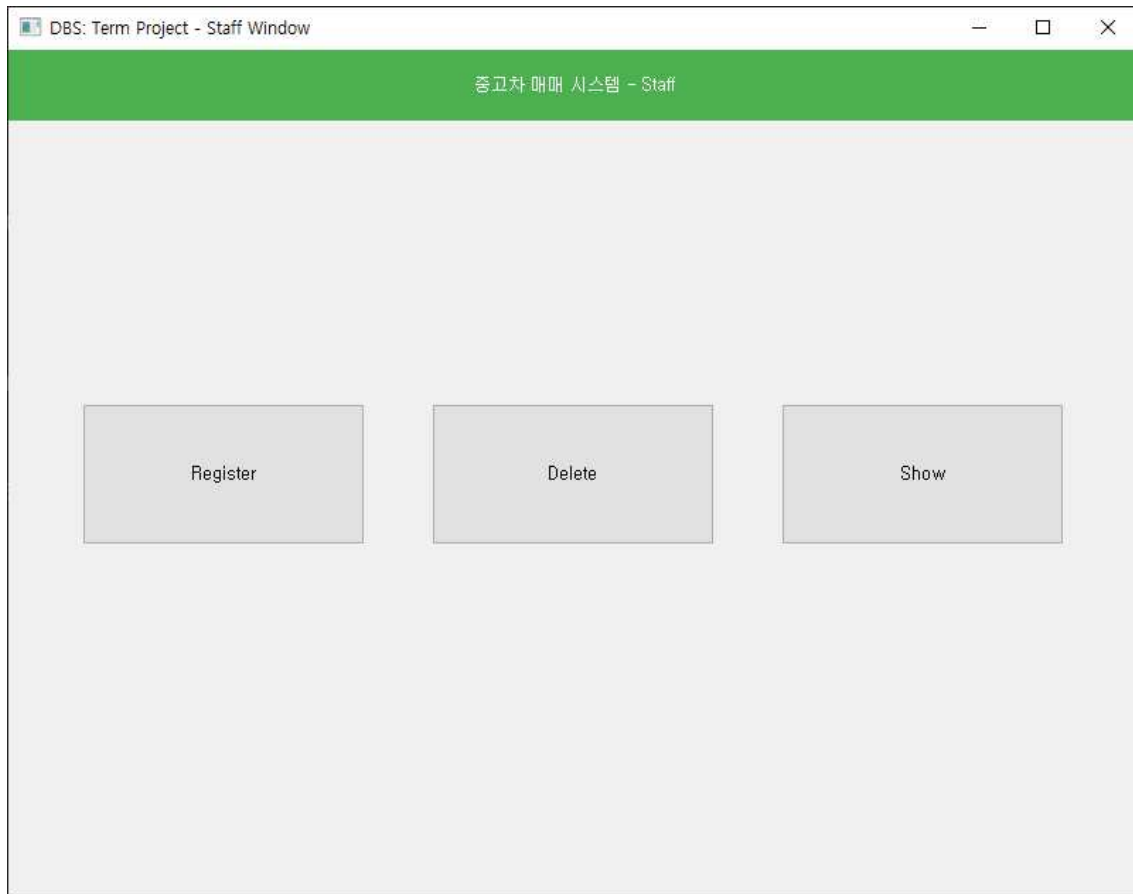
        # 해당 cid의 사고 여부를 확인
        cur.execute("SELECT * FROM Accident WHERE car_id = %s", (cid,))
        accident_data = cur.fetchall()

        if not accident_data:
            return False
        else:
            return True, accident_data

    except Exception as e:
        print(f"Error: {e}")
        return None

    finally:
        cur.close()
        remote.close()
```


4. Staff Window



```
def get_staff_name(sid):
    try:
        remote = mysql.connector.connect(
            host="192.168.56.101",
            user="goym",
            password="your_password",
            database="TermDB",
            port=4567
        )

        cur = remote.cursor()

        # Staff 테이블에서 sid에 해당하는 staff의 user_name을 가져옴
        cur.execute("SELECT user_name FROM Staff WHERE sid = %s", (sid,))
        staff_name = cur.fetchone()

        return staff_name[0] if staff_name else None

    except Exception as e:
        print(f"Error: {e}")
        return None

    finally:
        cur.close()
        remote.close()
```

중고차 매매 시스템 - Staff

중고차 등록

CID:

Age:

Model:

Distance:

Price:

Image:

등록

Show

```
def register_new_car(id, age, model, seller, distance, price, image):
    try:
        remote = mysql.connector.connect(
            host="192.168.56.101",
            user="goym",
            password="your_password",
            database="TermDB",
            port=4567
        )

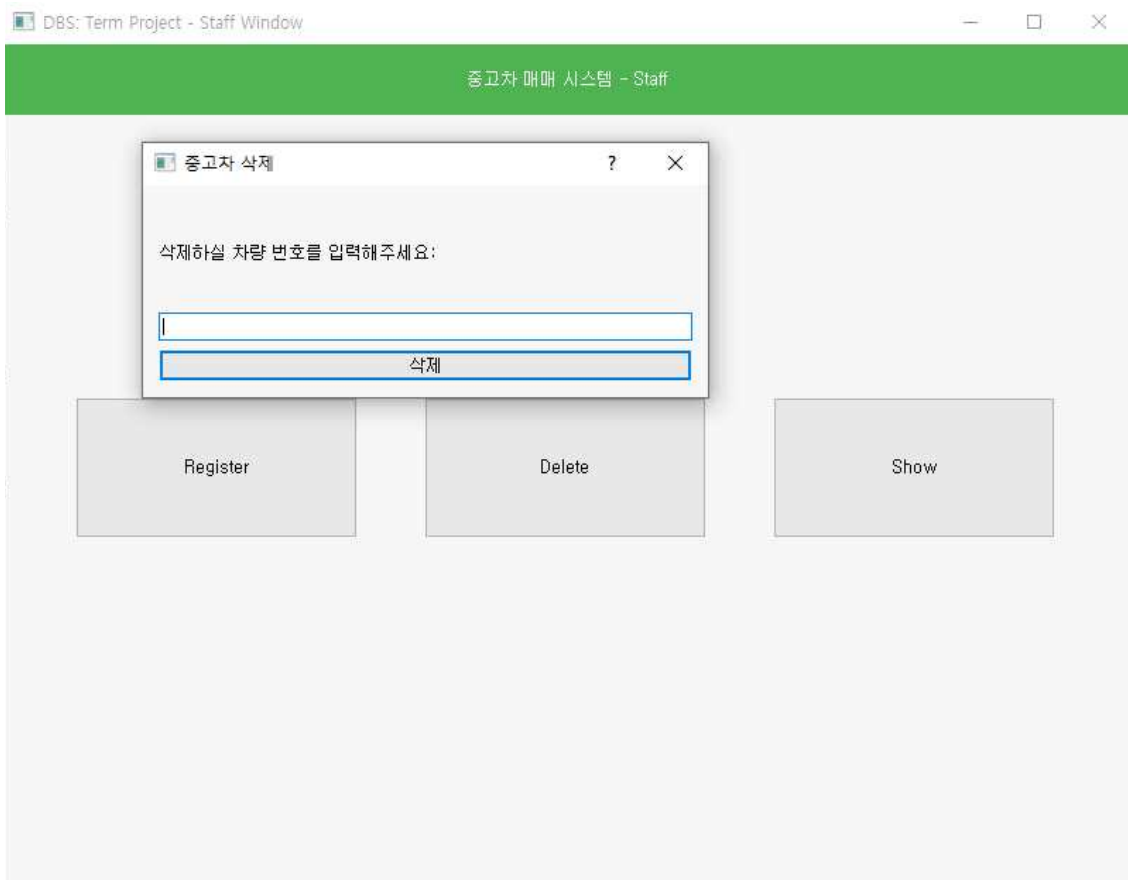
        cur = remote.cursor()

        check_query = f"SELECT cid FROM Car WHERE cid = '{id}'"
        cur.execute(check_query)
        existing_user = cur.fetchone()

        if existing_user:
            return False
        else:
            insert_query = f"INSERT INTO Car (cid, age, model, seller, distance, price, image)
                VALUES ('{id}', '{age}', '{model}', '{seller}', '{distance}', '{price}', '{image}')"
            cur.execute(insert_query)
            remote.commit()
            return True

    except Exception as e:
        print(f"Error: {e}")
        return False

    finally:
        cur.close()
        remote.close()
```



```
def delete_car(id):
    try:
        remote = mysql.connector.connect(
            host="192.168.56.101",
            user="goym",
            password="your_password",
            database="TermDB",
            port=4567
        )

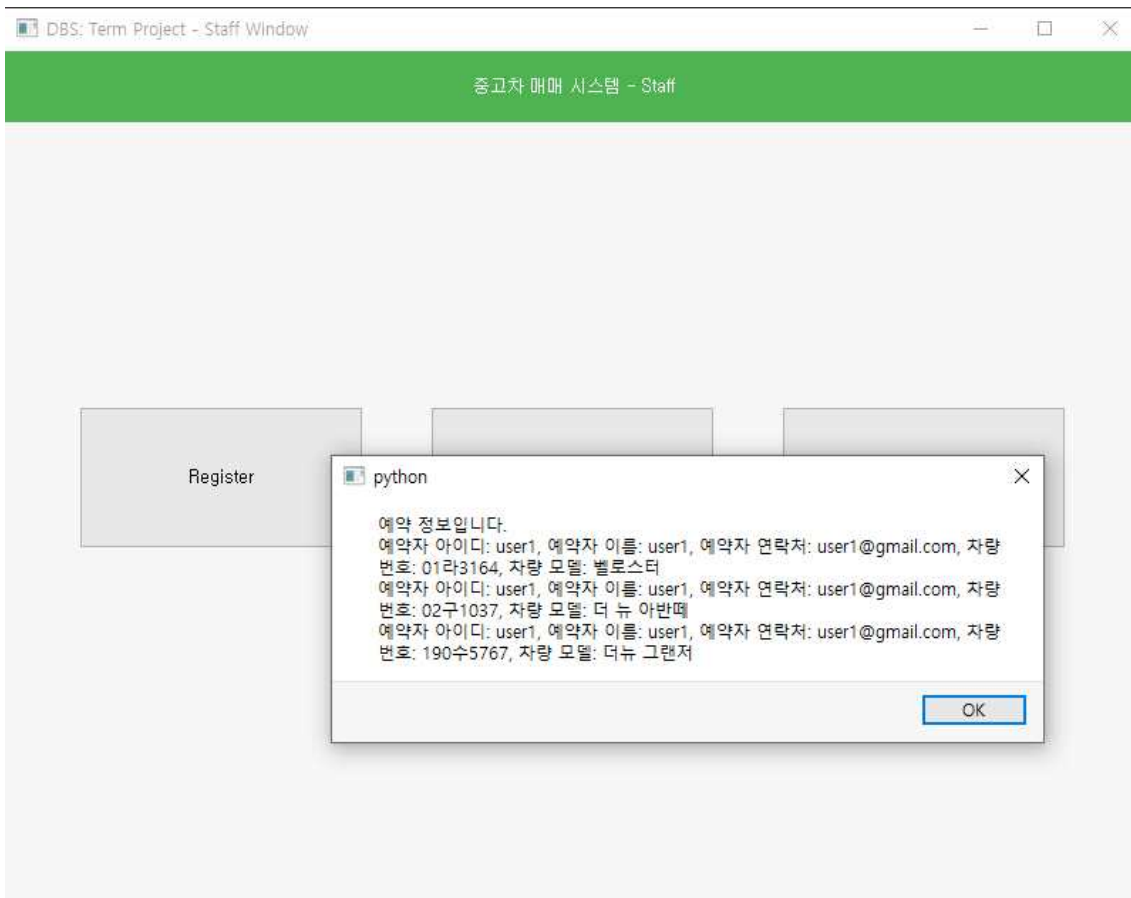
        cur = remote.cursor()

        check_query = f"SELECT * FROM Car WHERE cid = '{id}'"
        cur.execute(check_query)
        result = cur.fetchone()

        if result:
            delete_query = f"DELETE FROM Car WHERE cid = '{id}'"
            cur.execute(delete_query)
            remote.commit()
            print(f"차량 (CID: {id}) 삭제 성공!")
            return True
        else:
            print(f"차량 (CID: {id})가 존재하지 않습니다.")
            return False

    except Exception as e:
        print(f"Error: {e}")
        return False

    finally:
        cur.close()
        remote.close()
```



```
def show_reserver(sid):
    try:
        remote = mysql.connector.connect(
            host="192.168.56.101",
            user="goym",
            password="your_password",
            database="TermDB",
            port=4567
        )

        cur = remote.cursor(dictionary=True)

        # Staff의 sid와 일치하는 판매자가 Car 테이블에 있는 경우,
        cur.execute("""
            SELECT U.uid, U.user_name, U.email, C.cid, C.model
            FROM Car C
            JOIN User U ON C.buyer = U.uid
            WHERE C.seller = %s AND C.buyer IS NOT NULL
            """, (sid,))

        reservations = cur.fetchall()

        return reservations

    except Exception as e:
        print(f"Error: {e}")
        return None

    finally:
        cur.close()
        remote.close()
```