INHA UNIVERSITY

ICE4016 데이터베이스설계

< 설계 프로젝트 >

CAR_DEALER 데이타베이스 구현 및 Web응용 작성

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.
2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.
3. 나는 보고서의 내용을 조작하지 않겠습니다.
4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.
5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2022년 12 월 18 일

학부 정보통신공학
학년 3
성명 김가현
학번 12201853

1. 개요

- 해당 실습 또는 과제에서 구현해야 하는 목표 또는 개괄적인 설명 작성

2. 상세 설계 내용

- 해당 실습 또는 과제에서 구현한 내용에 대하여 중요한 부분의 코드 캡처 및 상세한 설명 작성
- 알고리즘 동작 과정 및 어떤 생각을 가지고 어떤 방식으로 구현했는지에 대하여

3. 실행 화면

- 프로그램을 실행시켰을 때의 결과 화면을 캡처하여 붙여넣기
- 실제로 실행했을 때와 결과가 같아야 하며, 해당 실습 또는 과제에서 요구하는 사항이 반드시 포함되어 있어야 함(없을 시 감점)

4. 결론

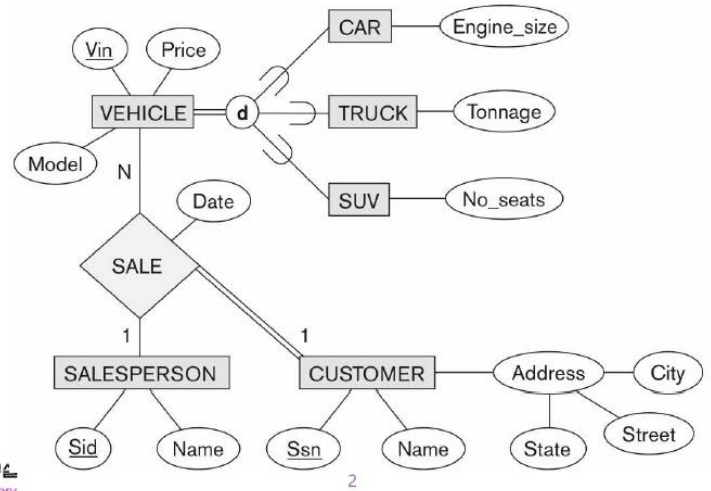
- 구현을 못했을 때의 문제점(어떤 점이 부족해서 구현을 못하였는지에 대한 분석)
- 구현하였을 경우 결과에 대한 분석

Final Project			
ID	12201853	**Name**	김가현

개요

○ 아래는 Car_Dealer에 대한 EER다이어그램이다.

○ 이 EER을 바탕으로 차량 판매시스템을 설계하라.



데이터 인텔리전스 연구실
Data Intelligence Laboratory

2

○ 요구 기능

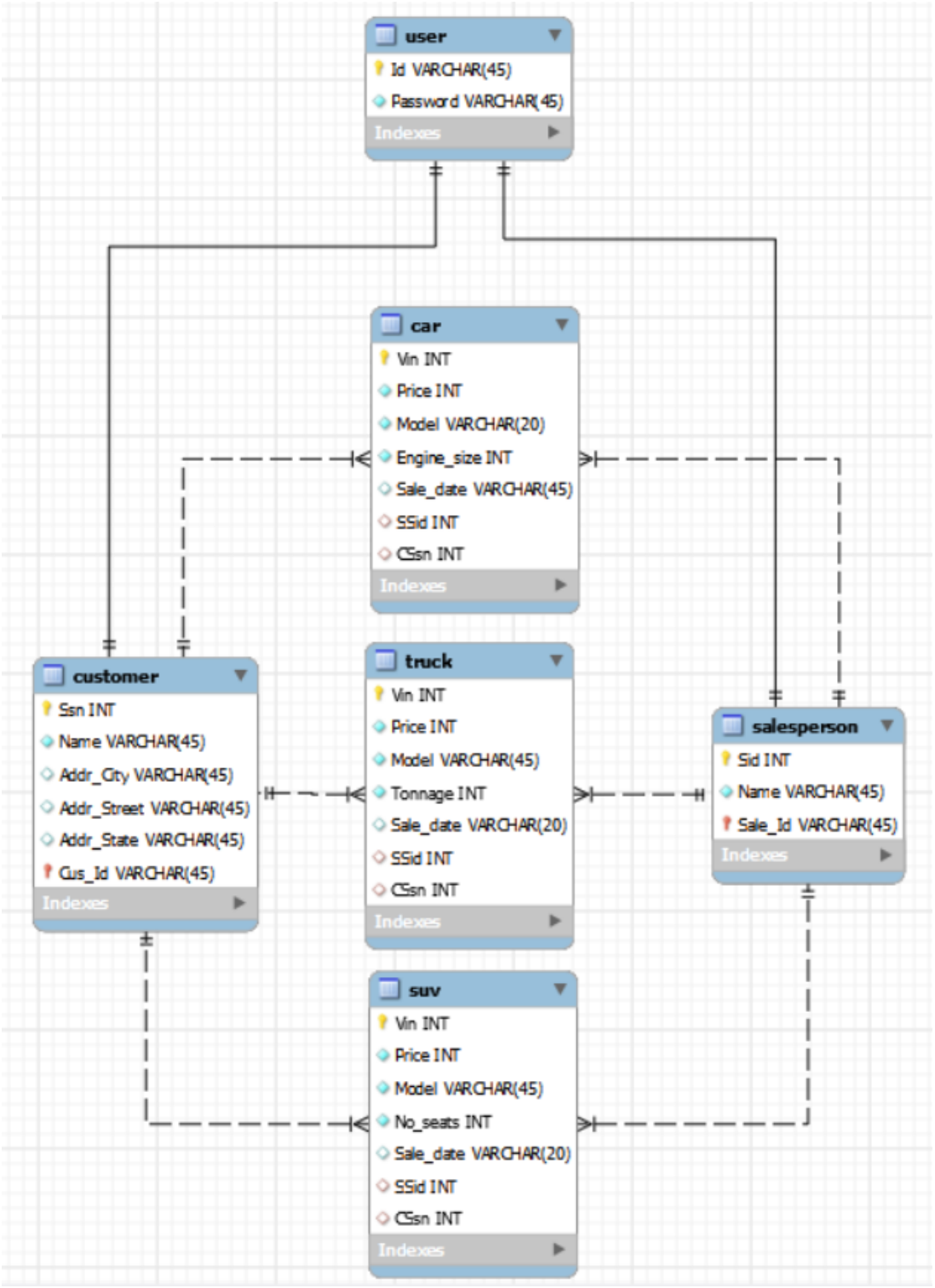
- 아래 두개의 페이지를 구성한다
  1. 차량 등록 및 판매를 관리하는 관리자 페이지
  2. 구매 예약, 예약조회 및 예약, 예약확인을 위한 사용자 페이지

1. 위 두 기능을 위한 요구사항을 재량껏 추가한 후 ERD를 완성하여 제시하라
2. 차량 데이터는 임의로 10만건 이상 만들어 DB에 입력한 상태로 시작
3. 관리자 페이지
   1. 차량 정보 입력/수정/삭제
   2. 차량 구매 예약 조회/수정/완료(판매완료, 판매실패)
      - 구매 예약후 차량 판매가 될때까지는 다른 사용자에게 그 차량은 검색되어서는 안됨
   3. 위 기능을 구현하기 위한 제약조건을 제시하라.
4. 사용자 페이지
   1. 차량 구매 예약하기, 예약조회, 예약취소 기능
   2. 위 기능을 구현하기 위한 제약조건, 색인, 트랜잭션 등을 제시하라.

Final Project			
ID	12201853	**Name**	김가현

상세 설계 내용

· ERD 작성



로그인 기능을 위해 user 테이블 추가

- 수업시간에 배운 내용을 바탕으로 disjoint 된 EER 을 그리고 있으면 vehicle 테이블은 생략하고, vehicle 테이블의 column 을 각 car, truck, suv 에 추가하였다.

i) Relationship

Tables	Relationship	Why?
Salesperson – car	**1:N Non-Identifying, 둘 모두 mandatory**	- 한 명의 salesperson 은 0 개 이상의 car 담당 가능 - 한 개의 car 에는 한 명의 salesperson 필요, 담당되지 않은 car 존재 가능
Salesperson – truck	**1:N Non-Identifying, 둘 모두 mandatory**	- 한 명의 salesperson 은 0 개 이상의 truck 담당 가능 - 한 개의 truck 에는 한 명의 salesperson 필요, 담당되지 않은 truck 존재 가능
Salesperson – suv	**1:N Non-Identifying, 둘 모두 mandatory**	- 한 명의 salesperson 은 0 개 이상의 suv 담당 가능 - 한 개의 suv 에는 한 명의 salesperson 필요, 담당되지 않은 suv 존재 가능
Customer – car	**1:N Non-Identifying, 둘 모두 mandatory**	- 한 명의 customer 는 0 개 이상의 car 구매 가능 - 한 개의 car 는 한 명의 customer 만 구매 가능, 구매하지 않은 car 존재 가능
Customer – truck	**1:N Non-Identifying, 둘 모두 mandatory**	- 한 명의 customer 는 0 개 이상의 truck 구매 가능 - 한 개의 truck 는 한 명의 customer 만 구매 가능, 구매하지 않은 truck 존재 가능
Customer – suv	**1:N Non-Identifying, 둘 모두 mandatory**	- 한 명의 customer 는 0 개 이상의 suv 구매 가능 - 한 개의 suv 는 한 명의 customer 만 구매 가능, 구매하지 않은 suv 존재 가능
Salesperson – user	**1:1 Identifying, 둘 모두 mandatory**	- 한 명의 salesperson 는 반드시 1 개의 로그인 위한 user 정보 필요 - 한 개의 user 정보는 반드시 salesperson 1 명을 가리킴
Customer – user	**1:1 Identifying, 둘 모두 mandatory**	- 한 명의 customer 는 반드시 1 개의 로그인 위한 user 정보 필요 - 한 개의 user 정보는 반드시 customer 1 명을 가리킴

ii) Table 별 Attributes

<user>

Attribute	Data Type	PK	NN	UQ	Why?
Id	**VARCHAR(45)**	●	●		- 값 중복 허용 X, Table 에 반드시 1 개 생성
Password	**VARCHAR(45)**		●		- 타 user 와 중복된 Password 존재 가능 - Password 는 로그인을 위해 반드시 존재해야 함

<customer>

Attribute	Data Type	PK	NN	UQ	Why?
Ssn	**INT**	●	●		- 값 중복 허용 X, Table 에 반드시 1 개 생성
Name	**VARCHAR(45)**		●		- customer 의 Name 은 반드시 존재 - 다른 customer 과 중복된 Name 가질 수 있음

Addr_City	**VARCHAR(45)**				- customer 의 Address 는 존재하지 않을 수 있음 - 다른 customer 와 같은 Address 에 거주 가능
Addr_Street	**VARCHAR(45)**				
Addr_State	**VARCHAR(45)**				
Cus_Id	**VARCHAR(45)**	●	●		- customer 에게는 로그인할 Cus_Id 를 반드시 가져야함

<salesperson>

Attribute	Data Type	PK	NN	UQ	Why?
Ssn	**INT**	●	●		- 값 중복 허용 X, Table 에 반드시 1 개 생성
Name	**VARCHAR(45)**		●		- salesperson 의 Name 은 반드시 존재 - 다른 salesperso 과 중복된 Name 가질 수 있음
Sale_Id	**VARCHAR(45)**	●	●		- salesperson 에게는 로그인할 Sale_Id 를 반드시 가져야함

<car>

Attribute	Data Type	PK	NN	UQ	Why?
Vin	**INT**	●	●		- 값 중복 허용 X, Table 에 반드시 1 개 생성
Price	**INT**		●		- 중복된 Price 가질 수 있음 - Price 가 없는 car 존재하지 않음
Model	**VARCHAR(20)**		●		- 같은 Model 을 여러 대 구비할 수 있음 - Model 명이 없는 car 존재하지 않음
Engine_size	**INT**		●		- 같은 Engine size 가진 car 존재 가능 - Engine size 를 측정할 수 없는 car 존재하지 X
Sale_date	**VARCHAR(45)**				- 같은 sale date 에 팔린 car 존재 가능 - 아직 팔리지 않은 car 존재 가능
SSid	**INT**				- 아직 salesperson 이 담당하지 않은 car 존재 0 - salesperson 은 여러 대의 car 담당 가능
CSsn	**INT**				- 아직 customer 가 배정되지 않은 car 존재 0 - customer 는 여러 대의 car 구매 가능

<truck>

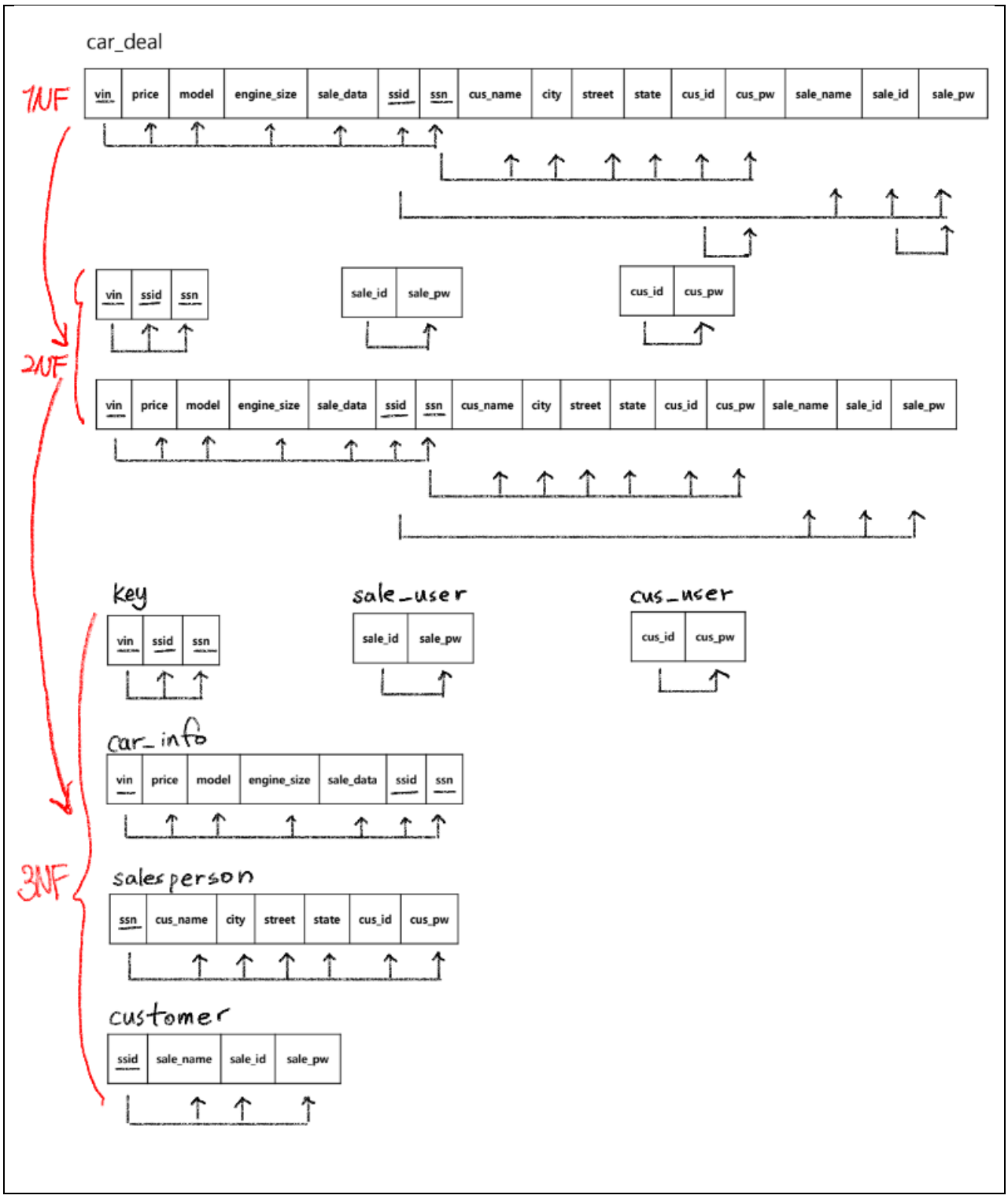
Attribute	Data Type	PK	NN	UQ	Why?
Vin	**INT**	●	●		- 값 중복 허용 X, Table 에 반드시 1 개 생성
Price	**INT**		●		- 중복된 Price 가질 수 있음 - Price 가 없는 truck 존재하지 않음
Model	**VARCHAR(20)**		●		- 같은 Model 을 여러 대 구비할 수 있음 - Model 명이 없는 truck 존재하지 않음
Tonnage	**INT**		●		- 같은 Tonnage 가진 truck 존재 가능 - Tonnage 를 측정할 수 없는 truck 존재하지 X

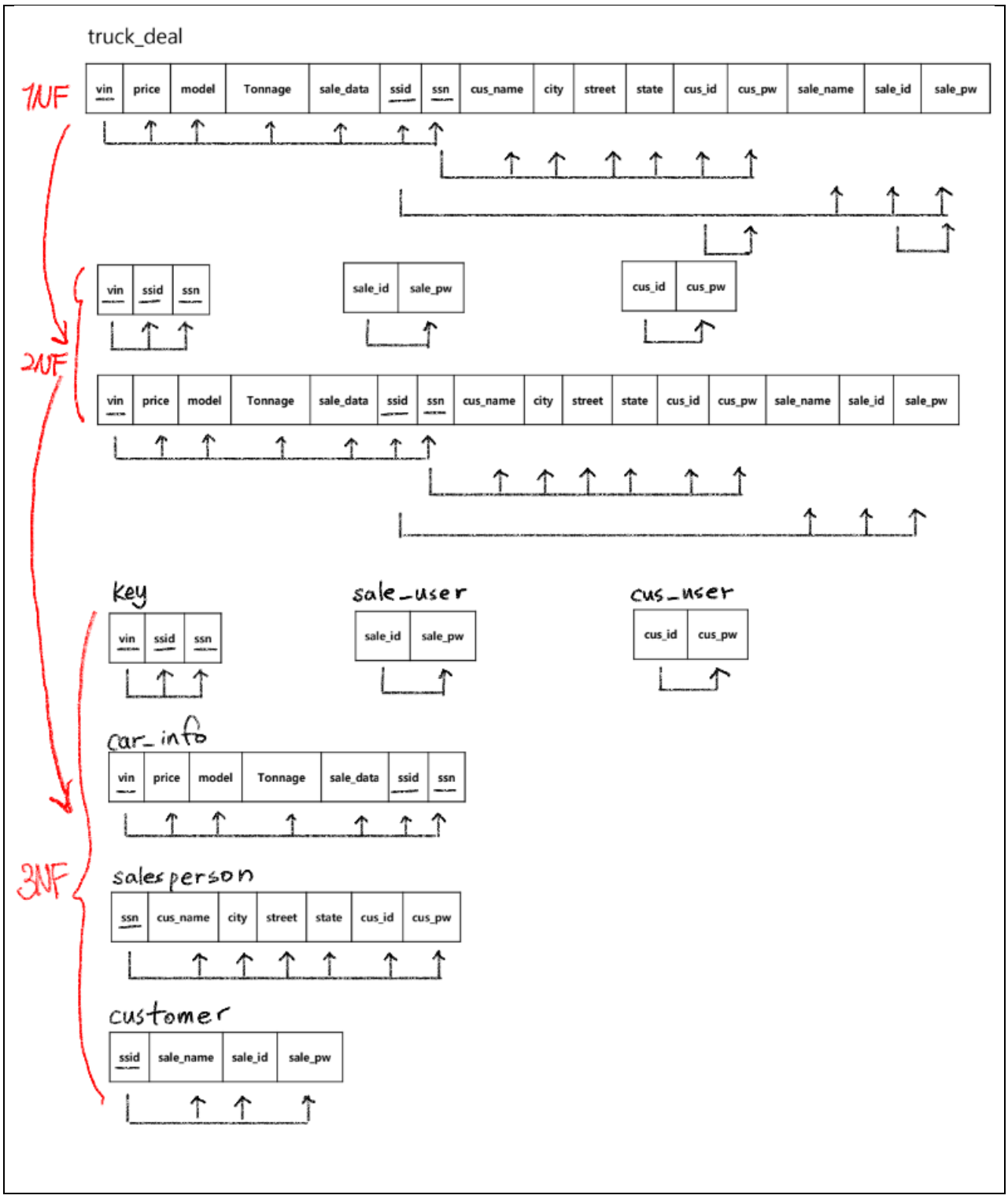
Sale_date	**VARCHAR(45)**				- 같은 sale date 에 팔린 truck 존재 가능 - 아직 팔리지 않은 truck 존재 가능
SSid	**INT**				- 아직 salesperson 이 담당하지 않은 truck 존재 0 - salesperson 은 여러 대의 truck 담당 가능
CSsn	**INT**				- 아직 customer 가 배정되지 않은 truck 존재 0 - customer 는 여러 대의 truck 구매 가능

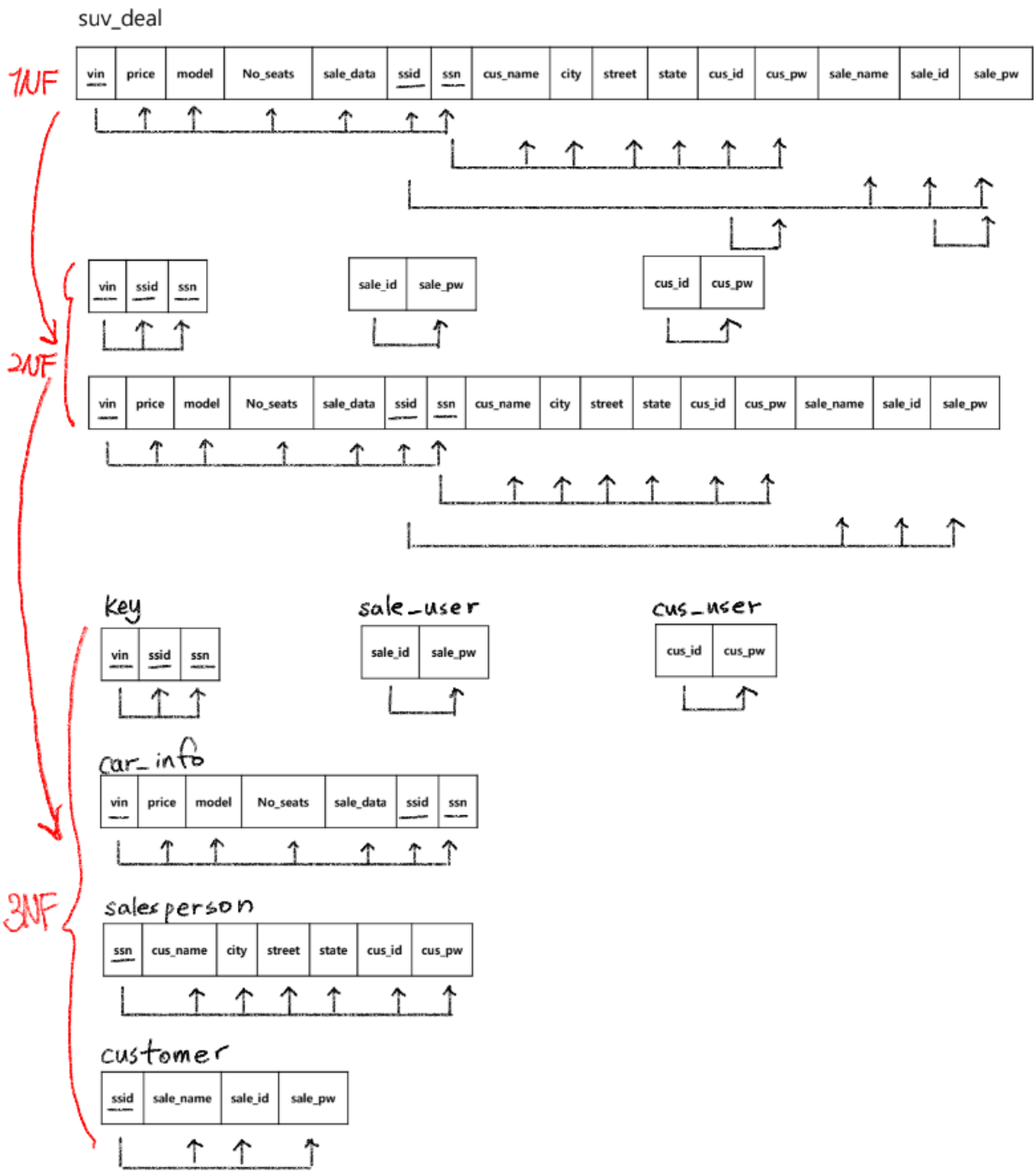
<suv>

Attribute	Data Type	PK	NN	UQ	Why?
Vin	**INT**	●	●		- 값 중복 허용 X, Table 에 반드시 1 개 생성
Price	**INT**		●		- 중복된 Price 가질 수 있음 - Price 가 없는 suv 존재하지 않음
Model	**VARCHAR(20)**		●		- 같은 Model 을 여러 대 구비할 수 있음 - Model 명이 없는 suv 존재하지 않음
No_seats	**INT**		●		- 같은 No seats 가진 car 존재 가능 - No seats 를 측정할 수 없는 suv 존재하지 X
Sale_date	**VARCHAR(45)**				- 같은 sale date 에 팔린 suv 존재 가능 - 아직 팔리지 않은 suv 존재 가능
SSid	**INT**				- 아직 salesperson 이 담당하지 않은 suv 존재 0 - salesperson 은 여러 대의 suv 담당 가능
CSsn	**INT**				- 아직 customer 가 배정되지 않은 suv 존재 0 - customer 는 여러 대의 suv 구매 가능

· 정규화 작업


car_deal
1NF
vin
price
model
engine_size
sale_data
ssid
ssn
cus_name
city
street
state
cus_id
cus_pw
sale_name
sale_id
sale_pw
2NF
vin
ssid
ssn
sale_id
sale_pw
cus_id
cus_pw
vin
price
model
engine_size
sale_data
ssid
ssn
cus_name
city
street
state
cus_id
cus_pw
sale_name
sale_id
sale_pw
3NF
key
vin
ssid
ssn
sale_user
sale_id
sale_pw
cus_user
cus_id
cus_pw
car_info
vin
price
model
engine_size
sale_data
ssid
ssn
salesperson
ssn
cus_name
city
street
state
cus_id
cus_pw
customer
ssid
sale_name
sale_id
sale_pw


truck_deal
1NF
vin
price
model
Tonnage
sale_data
ssid
ssn
cus_name
city
street
state
cus_id
cus_pw
sale_name
sale_id
sale_pw
2NF
vin
ssid
ssn
sale_id
sale_pw
cus_id
cus_pw
vin
price
model
Tonnage
sale_data
ssid
ssn
cus_name
city
street
state
cus_id
cus_pw
sale_name
sale_id
sale_pw
3NF
key
vin
ssid
ssn
sale_user
sale_id
sale_pw
cus_user
cus_id
cus_pw
car_info
vin
price
model
Tonnage
sale_data
ssid
ssn
salesperson
ssn
cus_name
city
street
state
cus_id
cus_pw
customer
ssid
sale_name
sale_id
sale_pw



각 vehicle 을 salesperson 과 customer 사이 deal 한 정보가 담긴 car_deal, truck_deal, suv_deal 테이블을 작성해주었다. 이는 부분 함수 종속 및 이행 함수 종속 모두를 담고 있어 1NF 에 속한다. 따라서 부분 함수 종속을 제거하여 2NF 를 얻고, 이행 함수 종속을 제거하여 3NF 를 얻어 위와 같이 정규화 하였다.

· 성능(색인) 최적화

```
 99983    INSERT INTO car(vin, price, model, engine_size) VALUES (33328, 5018, 'car18', 1);
 99984    INSERT INTO truck(vin, price, model, tonnage) VALUES (33328, 7018, 'truck18', 1);
 99985    INSERT INTO suv(vin, price, model, no_seats) VALUES (33328, 6018, 'suv18', 6);
 99986    INSERT INTO car(vin, price, model, engine_size) VALUES (33329, 5018, 'car18', 1);
 99987    INSERT INTO truck(vin, price, model, tonnage) VALUES (33329, 7018, 'truck18', 1);
 99988    INSERT INTO suv(vin, price, model, no_seats) VALUES (33329, 6018, 'suv18', 6);
 99989    INSERT INTO car(vin, price, model, engine_size) VALUES (33330, 5018, 'car18', 1);
 99990    INSERT INTO truck(vin, price, model, tonnage) VALUES (33330, 7018, 'truck18', 1);
 99991    INSERT INTO suv(vin, price, model, no_seats) VALUES (33330, 6018, 'suv18', 6);
 99992    INSERT INTO car(vin, price, model, engine_size) VALUES (33331, 5018, 'car18', 1);
 99993    INSERT INTO truck(vin, price, model, tonnage) VALUES (33331, 7018, 'truck18', 1);
 99994    INSERT INTO suv(vin, price, model, no_seats) VALUES (33331, 6018, 'suv18', 6);
 99995    INSERT INTO car(vin, price, model, engine_size) VALUES (33332, 5018, 'car18', 1);
 99996    INSERT INTO truck(vin, price, model, tonnage) VALUES (33332, 7018, 'truck18', 1);
 99997    INSERT INTO suv(vin, price, model, no_seats) VALUES (33332, 6018, 'suv18', 6);
 99998    INSERT INTO car(vin, price, model, engine_size) VALUES (33333, 5018, 'car18', 1);
 99999    INSERT INTO truck(vin, price, model, tonnage) VALUES (33333, 7018, 'truck18', 1);
100000    INSERT INTO suv(vin, price, model, no_seats) VALUES (33333, 6018, 'suv18', 6);
100001    INSERT INTO car(vin, price, model, engine_size) VALUES (33334, 5019, 'car19', 1);
100002    INSERT INTO truck(vin, price, model, tonnage) VALUES (33334, 7019, 'truck19', 1);
100003    INSERT INTO suv(vin, price, model, no_seats) VALUES (33334, 6019, 'suv19', 6);
```

car 33334 개, truck 33334 개, suv 33334 개 총 vehicle 10 만개의 데이터를 sql 문에 작성하여 터미널을 통해 insert 해주었다. 대량의 데이터를 다루기 위해 아래와 같이 색인(index) 작업을 진행해주었다. 색인은 Model 을 기준으로 하였다.

```
mysql> create index carmodel on car (model) using btree;
Query OK, 0 rows affected (0.21 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> create index truckmodel on truck (model) using btree;
Query OK, 0 rows affected (0.20 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> create index suvmodel on suv (model) using btree;
Query OK, 0 rows affected (0.15 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show index from car;
+-------+------------+---------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table | Non_unique | Key_name            | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-------+------------+---------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| car   |          0 | PRIMARY             |            1 | Vin         | A         |       32551 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| car   |          1 | fk_Car_Salesperson_idx |            1 | SSid        | A         |           2 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| car   |          1 | fk_Car_Customer1_idx |            1 | CSsn        | A         |           3 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| car   |          1 | carmodel            |            1 | Model       | A         |        2990 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+-------+------------+---------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
4 rows in set (0.00 sec)

mysql> show index from truck;
+-------+------------+-----------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table | Non_unique | Key_name              | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-------+------------+-----------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| truck |          0 | PRIMARY               |            1 | Vin         | A         |       33606 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| truck |          1 | fk_Truck_Salesperson1_idx |            1 | SSid        | A         |           2 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| truck |          1 | fk_Truck_Customer1_idx |            1 | CSsn        | A         |           2 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| truck |          1 | truckmodel            |            1 | Model       | A         |        3048 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+-------+------------+-----------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
4 rows in set (0.01 sec)

mysql> show index from suv;
+-------+------------+---------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table | Non_unique | Key_name            | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-------+------------+---------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| suv   |          0 | PRIMARY             |            1 | Vin         | A         |       32672 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| suv   |          1 | fk_Suv_Salesperson1_idx |            1 | SSid        | A         |           2 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| suv   |          1 | fk_Suv_Customer1_idx |            1 | CSsn        | A         |           2 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| suv   |          1 | suvmodel            |            1 | Model       | A         |        2990 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+-------+------------+---------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
4 rows in set (0.00 sec)
```

```
mysql> select count(vin) from car where model="car1";
+------------+
| count(vin) |
+------------+
|        384 |
+------------+
1 row in set (0.01 sec)

mysql> select count(vin) from truck where model="truck1";
+------------+
| count(vin) |
+------------+
|        384 |
+------------+
1 row in set (0.01 sec)

mysql> select count(vin) from suv where model="suv1";
+------------+
| count(vin) |
+------------+
|        384 |
+------------+
1 row in set (0.01 sec)
```

```
mysql> select count(vin) from car where model="car1";
+------------+
| count(vin) |
+------------+
|        384 |
+------------+
1 row in set (0.00 sec)

mysql> select count(vin) from truck where model="truck1";
+------------+
| count(vin) |
+------------+
|        384 |
+------------+
1 row in set (0.00 sec)

mysql> select count(vin) from suv where model="suv1";
+------------+
| count(vin) |
+------------+
|        384 |
+------------+
1 row in set (0.00 sec)
```

```
mysql> explain select count(vin) from car where model="car1";
+----+-------------+-------+------------+------+---------------+----------+---------+-------+------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key      | key_len | ref   | rows | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+----------+---------+-------+------+----------+-------------+
|  1 | SIMPLE      | car   | NULL       | ref  | carmodel      | carmodel | 62      | const |  384 |   100.00 | Using index |
+----+-------------+-------+------------+------+---------------+----------+---------+-------+------+----------+-------------+
1 row in set, 1 warning (0.01 sec)

mysql> explain select count(vin) from truck where model="truck1";
+----+-------------+-------+------------+------+---------------+------------+---------+-------+------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key        | key_len | ref   | rows | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+------------+---------+-------+------+----------+-------------+
|  1 | SIMPLE      | truck | NULL       | ref  | truckmodel    | truckmodel | 137     | const |  384 |   100.00 | Using index |
+----+-------------+-------+------------+------+---------------+------------+---------+-------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain select count(vin) from suv where model="suv1";
+----+-------------+-------+------------+------+---------------+----------+---------+-------+------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key      | key_len | ref   | rows | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+----------+---------+-------+------+----------+-------------+
|  1 | SIMPLE      | suv   | NULL       | ref  | suvmodel      | suvmodel | 137     | const |  384 |   100.00 | Using index |
+----+-------------+-------+------------+------+---------------+----------+---------+-------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

index 설정 이전과 이후 비교 결과, 검색 시간이 빨라진 것을 확인할 수 있었다. 이는 또한 사이트를 실행시킬 때, url 이동 속도가 빨라진 것을 vscode 터미널에서 확인할 수 있었다. 하지만 홈페이지에서 대량의 데이터를 모두 출력시키는 것은 한계가 있었다. price, engine_size 등을 기준으로 색인 작업을 늘려보았지만, 과한 index 는 느린 성능을 불러와 도움이 되지 않았다.

· Web

Web 은 12 주차에 진행했던 코드(인하대 포털)를 이용하였다. 따라서 이와 겹치는 부분 및 css/hbs 에 대한 자세한 설명은 생략하였다.

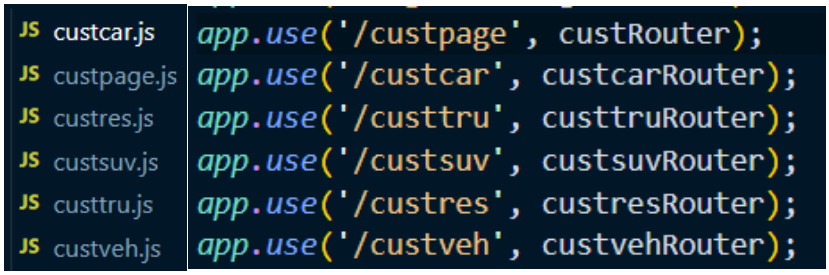
1. 로그인

```js
router.post('/', async (req, res) => {
    const vars = req.body;
    const users = await selectSql.getusers();
    let whoAmI = '';
    let Me = '';
    let checkLogin = false;

    users.map((user)=>{
        console.log(user.Id);
        if(vars.id===user.Id&&vars.password===user.Password){
            console.log('login success!');
            checkLogin=true;
            if(vars.id[0]==='c'){ //user.Id 첫 글자 c -> cust -> 사용자 아이디
                whoAmI='cust';
                Me = user.Id;
            }
            else{
                whoAmI='sale'; // salesperson
                Me = user.Id;
            }
        }
    })
```

```
    if(checkLogin&&whoAmI==='cust'){
        res.cookie('user', Me, {
            expires: new Date(Date.now() + 3600000), // ms 단위 (3600000: 1시간 유효)
            httpOnly: true
        })
        res.redirect('/custpage');
    }
    else if(checkLogin&&whoAmI==='sale'){
        res.cookie('user', Me, {
            expires: new Date(Date.now() + 3600000), // ms 단위 (3600000: 1시간 유효)
            httpOnly: true
        })
        res.redirect('/salepage');
    }
    else{
        console.log('login failed!');
        res.send("<script>alert('로그인에 실패했습니다.'); location.href='/';</script>")
    }
})
```

로그인 관리를 위해 cookie parser 를 이용하였다. user 의 id 는 cust 로 시작하면 customer 즉 사용자, sale 로 시작하면 salesperson 즉 관리자의 id 이다. 따라서 id 가 c 로 시작하면 사용자, 나머지는 관리자로 인식한다. 이때 잘못된 id 와 password 가 들어오면 로그인에 실패했다는 스크립트를 띄웠다.

2. 사용자



사용자 페이지에 해당하는 url

custpage : 사용자로 로그인 시 가장 처음 뜨는 페이지로, 사용자 홈페이지의 main 이 된다.

<custcar, custtru, custsuv>

```
var m="car0"

router.get('/', async function (req, res) {
    const car = await selectSql.getnotrescar(m);
    if (req.cookies.user) {
        res.render('custcar', {
            user: req.cookies.user,
            title: 'CAR 목록',
            car,
    })
    } else {
        res.render('/')
    }
})

router.post('/search', async (req, res) => {
    console.log('search router:', req.body.seaBtn);
    m=req.body.Model

    res.redirect('/custcar');
});

router.post('/', async (req, res) => {
    const vars = req.body;
    const data = {
        Vin: vars.Vin,
        Ssn: vars.Ssn
    }
    await updateSql.updatecarres(data);

    res.redirect('/custres');
});
```

```
var m="truck0"

router.get('/', async function (req, res) {
    const truck = await selectSql.getnotrestruck(m);
    if (req.cookies.user) {
        res.render('custtru', {
            user: req.cookies.user,
            title: 'TRUCK 목록',
            truck,
    })
    } else {
        res.render('/')
    }
})

router.post('/search', async (req, res) => {
    console.log('search router:', req.body.seaBtn);
    m=req.body.Model

    res.redirect('/custtru');
});

router.post('/', async (req, res) => {
    const vars = req.body;
    const data = {
        Vin: vars.Vin,
        Ssn: vars.Ssn
    }
    await updateSql.updatetruckres(data);

    res.redirect('/custres');
});
```

```
var m="suv0"

router.get('/', async function (req, res) {
    const suv = await selectSql.getnotressuv(m);
    if (req.cookies.user) {
        res.render('custsuv', {
            user: req.cookies.user,
            title: 'SUV 목록',
            suv,
    })
    } else {
        res.render('/')
    }
})

router.post('/search', async (req, res) => {
    console.log('search router:', req.body.seaBtn);
    m=req.body.Model

    res.redirect('/custsuv');
});

router.post('/', async (req, res) => {
    const vars = req.body;
    const data = {
        Vin: vars.Vin,
        Ssn: vars.Ssn
    }
    await updateSql.updatesuvres(data);

    res.redirect('/custres');
});
```

```
//예약 완된 중고차 : 딜러, 예약자 모두 NULL
getnotrescar: async (model) => {
    const { k } = require('C:/Users/aille/Desktop/DBfinal/src/routes/login');
    const sql = `select car.Vin, car.Price, car.Model, car.Engine_size, customer.Ssn from car, customer where customer.cus_id="${k}" and car.SSid is NULL and car.CSsn is NULL and car.model="${model}"`;
    const [result] = await promisePool.query(sql);

    return result;
},
getnotrestruck: async (model) => {
    const { k } = require('C:/Users/aille/Desktop/DBfinal/src/routes/login');
    const sql = `select truck.Vin, truck.Price, truck.Model, truck.Tonnage, customer.Ssn from truck, customer where customer.cus_id="${k}" and truck.SSid is NULL and truck.CSsn is NULL and truck.model="${model}`;
    const [result] = await promisePool.query(sql);

    return result;
},
getnotressuv: async (model) => {
    const { k } = require('C:/Users/aille/Desktop/DBfinal/src/routes/login');
    const sql = `select suv.Vin, suv.Price, suv.Model, suv.No_seats, customer.Ssn from suv, customer where customer.cus_id="${k}" and suv.SSid is NULL and suv.CSsn is NULL and suv.model="${model}"`;
    const [result] = await promisePool.query(sql);

    return result;
},
```

```
// 구매 예약하기 : CSsn 등록
updatecarres: async (data) => {
  console.log(data);
  const sql = `update car set CSsn="${data.Ssn}" where Vin="${data.Vin}"`;
  await promisePool.query(sql);
},

updatetruckres: async (data) => {
    console.log(data);
    const sql = `update truck set CSsn="${data.Ssn}" where Vin="${data.Vin}"`;
    await promisePool.query(sql);
},

updatesuvres: async (data) => {
    console.log(data);
    const sql = `update suv set CSsn="${data.Ssn}" where Vin="${data.Vin}"`;
    await promisePool.query(sql);
},
```

```
const k=vars.id;
module.exports={
    k
};
```

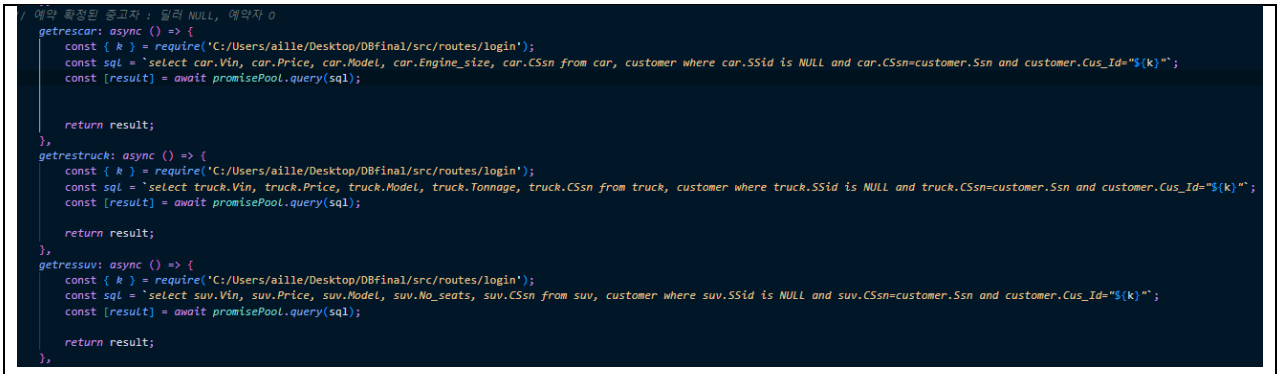
k: ID 저장한 변수

custcar, custtru, custsuv 는 각각 차량 구매를 예약할 수 있는 페이지이며, select 문을 통해 예약할 수 있는 차량 목록 및 정보가 뜨고 그 옆에는 예약할 수 있는 버튼(updBtn)을 생성하였다. 이때 버튼을 누르면 update 문을 통해 vehicle 의 CSsn 즉 customer ssn 자리를 로그인한 사용자의 ssn(Not NULL)으로 변경한다.

<custres>

```
router.get('/', async function (req, res) {
    const car = await selectSql.getrescar();
    const truck = await selectSql.getrestruck();
    const suv = await selectSql.getressuv();
    if (req.cookies.user) {
        res.render('custres', {
            user: req.cookies.user,
            title: '예약 조회/취소',
            car,
            truck,
            suv,
    })
    } else {
        res.render('/')
    }
})

router.post('/', async (req, res) => {
    const vars = req.body;
    const data = {
        Vin: vars.Vin,
        Model: vars.Model
    }
    await updateSql.cancelcarres(data);
    await updateSql.canceltruckres(data);
    await updateSql.cancelsuvres(data);

    res.redirect('/custres');
});
```

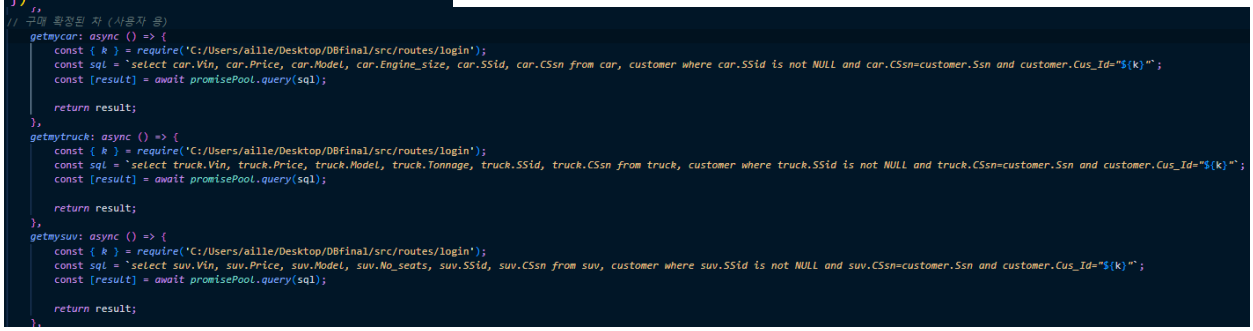
```
// 예약 취소 : 예약자 -> NULL
cancelcarres: async (data) => {
    console.log(data);
    const sql = `update car set CSsn=NULL where Vin="${data.Vin}" and Model="${data.Model}"`;
    await promisePool.query(sql);
},
canceltruckres: async (data) => {
    console.log(data);
    const sql = `update truck set CSsn=NULL where Vin="${data.Vin}" and Model="${data.Model}"`;
    await promisePool.query(sql);
},
cancelsuvres: async (data) => {
    console.log(data);
    const sql = `update suv set CSsn=NULL where Vin="${data.Vin}" and Model="${data.Model}"`;
    await promisePool.query(sql);
},
```



custres 는 사용자가 예약한 차량에 대해 예약을 조회하고, 원하지 않으면 취소할 수 있는 페이지이다. 세 vehicle 의 예약 목록을 모두 보여주고, select 문을 통해 본인의 예약 정보만 볼 수 있다. 예약 정보 옆에는 취소 버튼을 생성하여, 버튼을 누르면 update 문을 통해 vehicle 의 CSsn 즉 customer ssn 자리를 NULL 로 변경하여 예약자가 없음을 나타내도록 한다. 이때 변경하면 본인은 물론 다른 사용자까지 예약하기 페이지에서 그 차량을 다시 볼 수 있다.

<custveh>

```
router.get('/', async function (req, res) {
    const car = await selectSql.getmycar();
    const truck = await selectSql.getmytruck();
    const suv = await selectSql.getmysuv();
    if (req.cookies.user) {
        res.render('custveh', {
            user: req.cookies.user,
            title: 'My Vehicle',
            car,
            truck,
            suv,
    })
    } else {
        res.render('/')
    }
})
```



custveh 는 사용자가 구매를 완료한 차량의 목록을 볼 수 있는 페이지이다. 이는 select 문을 통해 로그인한 id 와 vehicle 의 cssn(customer ssn)가 일치하고, ssid(salesperson id)도 NULL 이 아닌 vehicle 데이터를 출력하여 본인이 구매한 차량 정보만 볼 수 있다.

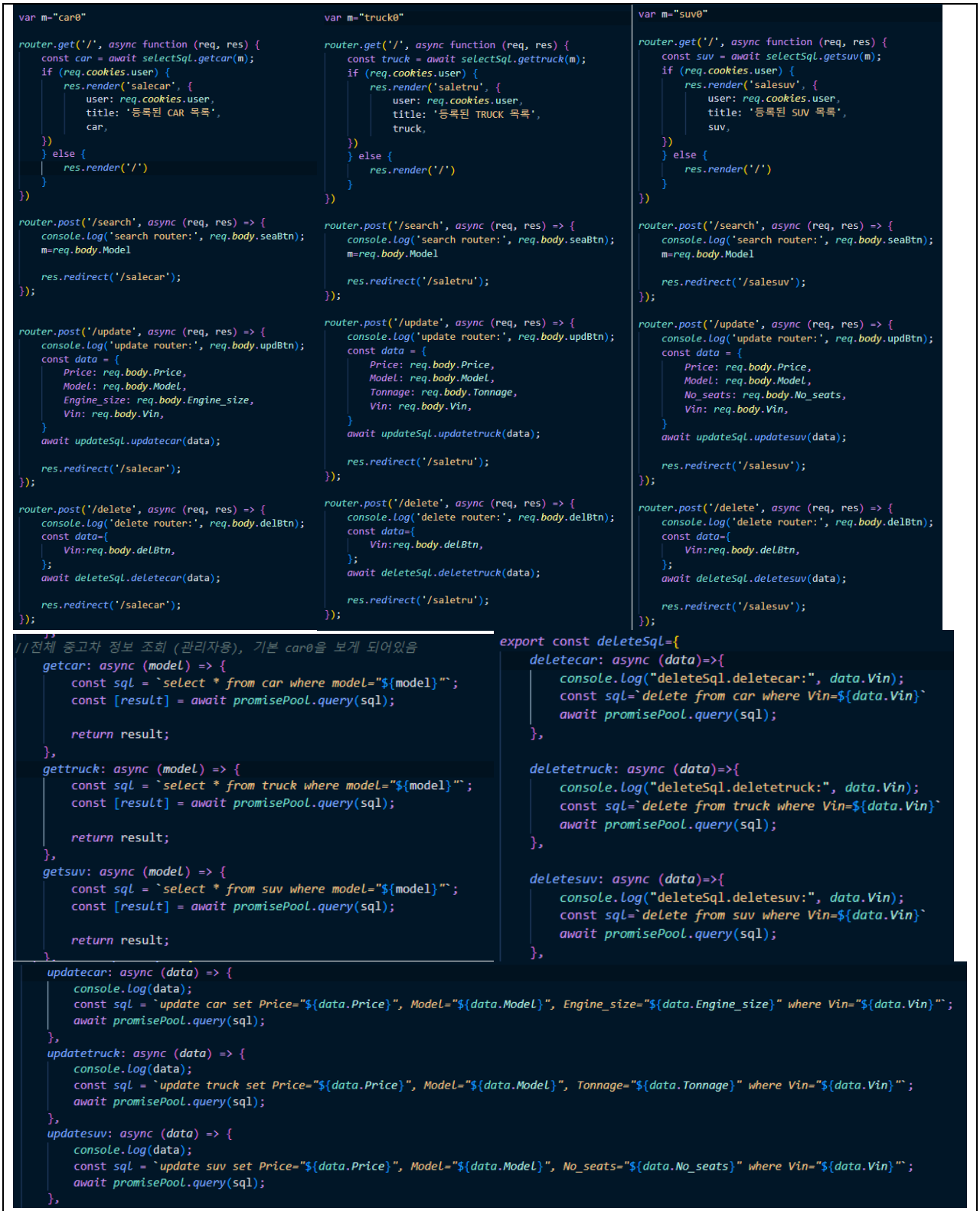
.3. 관리자

```
app.use('/salepage', saleRouter);
app.use('/salecar', salecarRouter);
app.use('/saletru', saletruRouter);
app.use('/salesuv', salesuvRouter);
app.use('/saleres', saleresRouter);
```

관리자 페이지에 해당하는 url

salepage: 사용자로 로그인 시 가장 처음 뜨는 페이지로, 사용자 홈페이지의 main 이 된다.

<salecar, saletru, salesuv>



```
var m="car0"

router.get('/', async function (req, res) {
    const car = await selectSql.getcar(m);
    if (req.cookies.user) {
        res.render('salecar', {
            user: req.cookies.user,
            title: '등록된 CAR 목록',
            car,
    })
    } else {
        res.render('/')
    }
})

router.post('/search', async (req, res) => {
    console.log('search router:', req.body.seaBtn);
    m=req.body.Model

    res.redirect('/salecar');
});


router.post('/update', async (req, res) => {
    console.log('update router:', req.body.updBtn);
    const data = {
        Price: req.body.Price,
        Model: req.body.Model,
        Engine_size: req.body.Engine_size,
        Vin: req.body.Vin,
    }
    await updateSql.updatecar(data);

    res.redirect('/salecar');
});

router.post('/delete', async (req, res) => {
    console.log('delete router:', req.body.delBtn);
    const data={
        Vin:req.body.delBtn,
    };
    await deleteSql.deletecar(data);

    res.redirect('/salecar');
});
```

```
var m="truck0"

router.get('/', async function (req, res) {
    const truck = await selectSql.gettruck(m);
    if (req.cookies.user) {
        res.render('saletru', {
            user: req.cookies.user,
            title: '등록된 TRUCK 목록',
            truck,
    })
    } else {
        res.render('/')
    }
})

router.post('/search', async (req, res) => {
    console.log('search router:', req.body.seaBtn);
    m=req.body.Model

    res.redirect('/saletru');
});

router.post('/update', async (req, res) => {
    console.log('update router:', req.body.updBtn);
    const data = {
        Price: req.body.Price,
        Model: req.body.Model,
        Tonnage: req.body.Tonnage,
        Vin: req.body.Vin,
    }
    await updateSql.updatetruck(data);

    res.redirect('/saletru');
});

router.post('/delete', async (req, res) => {
    console.log('delete router:', req.body.delBtn);
    const data={
        Vin:req.body.delBtn,
    };
    await deleteSql.deletetruck(data);

    res.redirect('/saletru');
});
```

```
var m="suv0"

router.get('/', async function (req, res) {
    const suv = await selectSql.getsuv(m);
    if (req.cookies.user) {
        res.render('salesuv', {
            user: req.cookies.user,
            title: '등록된 SUV 목록',
            suv,
    })
    } else {
        res.render('/')
    }
})

router.post('/search', async (req, res) => {
    console.log('search router:', req.body.seaBtn);
    m=req.body.Model

    res.redirect('/salesuv');
});

router.post('/update', async (req, res) => {
    console.log('update router:', req.body.updBtn);
    const data = {
        Price: req.body.Price,
        Model: req.body.Model,
        No_seats: req.body.No_seats,
        Vin: req.body.Vin,
    }
    await updateSql.updatesuv(data);

    res.redirect('/salesuv');
});

router.post('/delete', async (req, res) => {
    console.log('delete router:', req.body.delBtn);
    const data={
        Vin:req.body.delBtn,
    };
    await deleteSql.deletesuv(data);

    res.redirect('/salesuv');
});
```

```
//전체 중고차 정보 조회 (관리자용), 기본 car0을 보게 되어있음
    getcar: async (model) => {
        const sql = `select * from car where model="${model}"`;
        const [result] = await promisePool.query(sql);

        return result;
    },
    gettruck: async (model) => {
        const sql = `select * from truck where model="${model}"`;
        const [result] = await promisePool.query(sql);

        return result;
    },
    getsuv: async (model) => {
        const sql = `select * from suv where model="${model}"`;
        const [result] = await promisePool.query(sql);

        return result;
    },
```

```
export const deleteSql={
    deletecar: async (data)=>{
        console.log("deleteSql.deletecar:", data.Vin);
        const sql=`delete from car where Vin=${data.Vin}`
        await promisePool.query(sql);
    },

    deletetruck: async (data)=>{
        console.log("deleteSql.deletetruck:", data.Vin);
        const sql=`delete from truck where Vin=${data.Vin}`
        await promisePool.query(sql);
    },

    deletesuv: async (data)=>{
        console.log("deleteSql.deletesuv:", data.Vin);
        const sql=`delete from suv where Vin=${data.Vin}`
        await promisePool.query(sql);
    },
```

```
    updatecar: async (data) => {
        console.log(data);
        const sql = `update car set Price="${data.Price}", Model="${data.Model}", Engine_size="${data.Engine_size}" where Vin="${data.Vin}"`;
        await promisePool.query(sql);
    },
    updatetruck: async (data) => {
        console.log(data);
        const sql = `update truck set Price="${data.Price}", Model="${data.Model}", Tonnage="${data.Tonnage}" where Vin="${data.Vin}"`;
        await promisePool.query(sql);
    },
    updatesuv: async (data) => {
        console.log(data);
        const sql = `update suv set Price="${data.Price}", Model="${data.Model}", No_seats="${data.No_seats}" where Vin="${data.Vin}"`;
        await promisePool.query(sql);
    },
```

salecar, saletru, salesuv 는 각각 차량 정보를 조회할 수 있으며, 정보 옆에 수정 버튼과 삭제 버튼으로 차량 정보를 수정 또는 삭제를 할 수 있는 페이지이다. 약 3 만 개의 데이터를 한 번에 출력하는 것에는 무리가 있다고 생각하여, 차량 정보 표 위에 모델명을 검색할 수 있는 기능을 구현하였다. 모델명을 입력하고 검색 버튼을 누르면, select 문을 통해 테이블 안에 그 모델명을 가진 차량 정보를 출력한다. 차량 정보를 수정하고 수정 버튼을 누르면 update 문을 통해 수정한 차량의 vin 에 해당하는 차량 정보를 수정한다. update 의 경우, price, model, engine_size(tonnage, no_seats)만 수정 가능하도록 구현하였다. 삭제 버튼을 누르면 delete 문을 통해 그 차량 정보를 삭제한다.

<saleres>

```
router.get('/', async function (req, res) {
    const car = await selectSql.getrescar2();
    const truck = await selectSql.getrestruck2();
    const suv = await selectSql.getressuv2();
    const salecar = await selectSql.getsalecar();
    const saletruck = await selectSql.getsaletruck();
    const salesuv = await selectSql.getsalesuv();

    if (req.cookies.user) {
        res.render('saleres', {
            user: req.cookies.user,
            title1: '예약 현황',
            car,
            truck,
            suv,
            title2: '판매 완료 현황',
            salecar,
            saletruck,
            salesuv
    })
    } else {
        res.render('/')
    }
})

router.post('/succ', async (req, res) => {
    console.log('update router:', req.body.updBtn);
    const data = {
        Sid: req.body.Sid,
        Vin: req.body.Vin,
        Model: req.body.Model,
    }
    await updateSql.carsalesucc(data);
    await updateSql.trucksalesucc(data);
    await updateSql.suvsalesucc(data);

    res.redirect('/saleres');
});

router.post('/fail', async (req, res) => {
    console.log('update router:', req.body.updBtn);
    const data = {
        Vin: req.body.Vin,
        Model: req.body.Model,
    }
    await updateSql.cancelcarres(data);
    await updateSql.canceltruckres(data);
    await updateSql.cancelsuvres(data);

    res.redirect('/saleres');
});
```

관리자 예약 실패 = 사용자 예약 취소 (사용자 NULL) -> sql 문 같음

```
function getToday(){ // 판매날짜 가져오기 위한 오늘 날짜
    var date = new Date();
    var year = date.getFullYear();
    var month = ("0" + (1 + date.getMonth())).slice(-2);
    var day = ("0" + date.getDate()).slice(-2);

    return year + "-" + month + "-" + day;
}
```

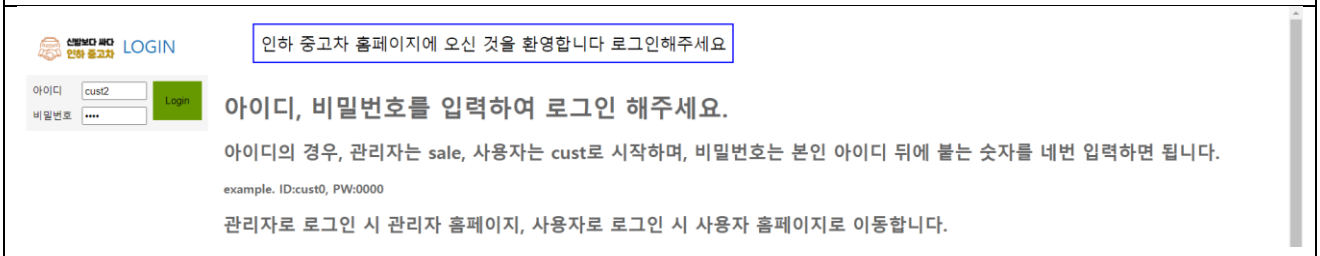
판매 날짜 update 위하여 오늘의 날짜를 return 하는 함수 작성

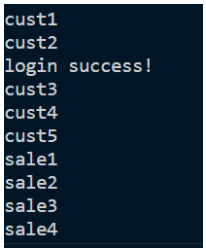
```
// 판매 성공 : 딜러 샘킴
    carsalesucc: async (data) => {
        const today=getToday();
        console.log(data);
        const sql = `update car set Sale_date="${today}", SSid="${data.Sid}" where Vin="${data.Vin}" and Model="${data.Model}"`;
        await promisePool.query(sql);
    },
    trucksalesucc: async (data) => {
        const today=getToday();
        console.log(data);
        const sql = `update truck set Sale_date="${today}", SSid="${data.Sid}" where Vin="${data.Vin}" and Model="${data.Model}"`;
        await promisePool.query(sql);
    },
    suvsalesucc: async (data) => {
        const today=getToday();
        console.log(data);
        const sql = `update suv set Sale_date="${today}", SSid="${data.Sid}" where Vin="${data.Vin}" and Model="${data.Model}"`;
        await promisePool.query(sql);
    },
// 예약 확정된 중고차(관리자용) : 딜러 NULL, 예약자 O
getrescar2: async () => {
    const { k } = require('C:/Users/aille/Desktop/DBfinal/src/routes/login');
    const sql = `select car.Vin, car.Price, car.Model, car.Engine_size, car.CSsn, salesperson.Sid from car, salesperson where salesperson.Sale_Id="${k}" and car.SSid is NULL and car.CSsn is not NULL`;
    const [result] = await promisePool.query(sql);

    return result;
},
getrestruck2: async () => {
    const { k } = require('C:/Users/aille/Desktop/DBfinal/src/routes/login');
    const sql = `select truck.Vin, truck.Price, truck.Model, truck.Tonnage, truck.CSsn, salesperson.Sid from truck, salesperson where salesperson.Sale_Id="${k}" and truck.SSid is NULL and truck.CSsn is not NULL`;
    const [result] = await promisePool.query(sql);

    return result;
},
getressuv2: async () => {
    const { k } = require('C:/Users/aille/Desktop/DBfinal/src/routes/login');
    const sql = `select suv.Vin, suv.Price, suv.Model, suv.No_seats, suv.CSsn, salesperson.Sid from suv, salesperson where salesperson.Sale_Id="${k}" and suv.SSid is NULL and suv.CSsn is not NULL`;
    const [result] = await promisePool.query(sql);

    return result;
},
getsalecar: async () => {
    const sql = `select * from car where SSid is not NULL and CSsn is not NULL`;
    const [result] = await promisePool.query(sql);

    return result;
},
getsaletruck: async () => {
    const sql = `select * from truck where SSid is not NULL and CSsn is not NULL`;
    const [result] = await promisePool.query(sql);

    return result;
},
getsalesuv: async () => {
    const sql = `select * from suv where SSid is not NULL and CSsn is not NULL`;
    const [result] = await promisePool.query(sql);

    return result;
},
```

saleres 는 사용자가 예약한 현황을 볼 수 있으며 판매 완료 또는 실패를 정할 수 있고, 판매가 완료된 차량도 조회할 수 있다. 예약 정보에서 판매 완료 버튼을 누르면, update 문을 통해 판매 날짜, 관리자의 sid 를 차량 정보에 입력한다. 이를 통해 차량의 모든 attribute 에 NULL 이 없는 즉 판매가 완료된 것이며, 사용자는 구매가 확정되는 것이다. 반대로 판매 실패 버튼을 누르면, 차량 정보에 추가되었던 예약자의 ssn 이 NULL 로 update 된다. 판매가 완료된 차량 정보는 ssid 와 cssn 이 NULL 이 아닌 차량 정보만 출력하도록 하는 select 문을 통해 조회할 수 있다.

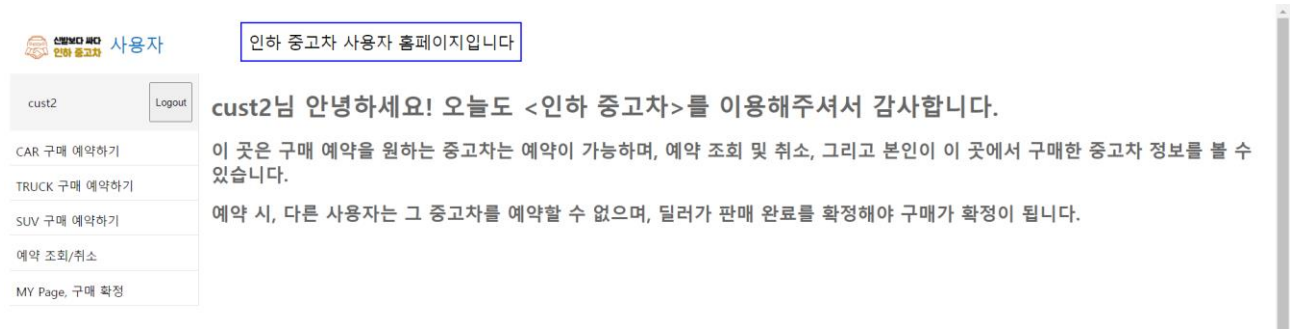
Final Project			
ID	12201853	**Name**	김가현

실행 화면



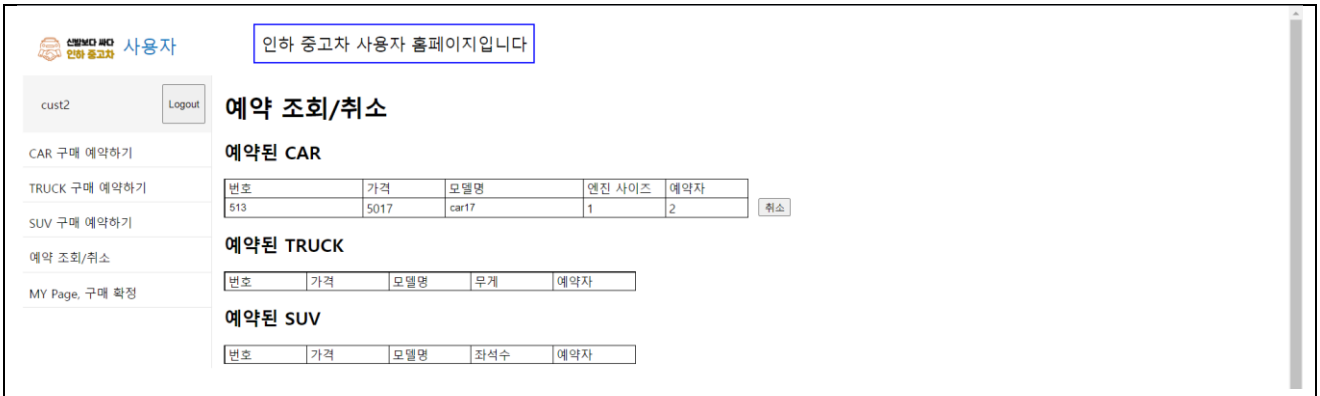


cust2(사용자)로 로그인, 비밀번호 2222

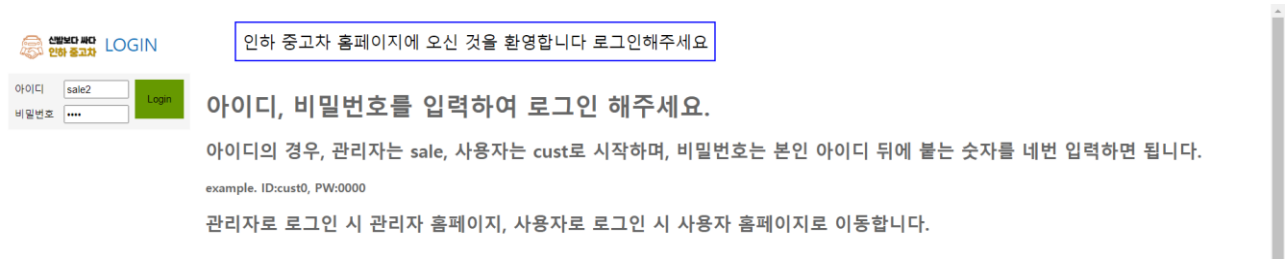


사용자 메인 페이지인 /custpage

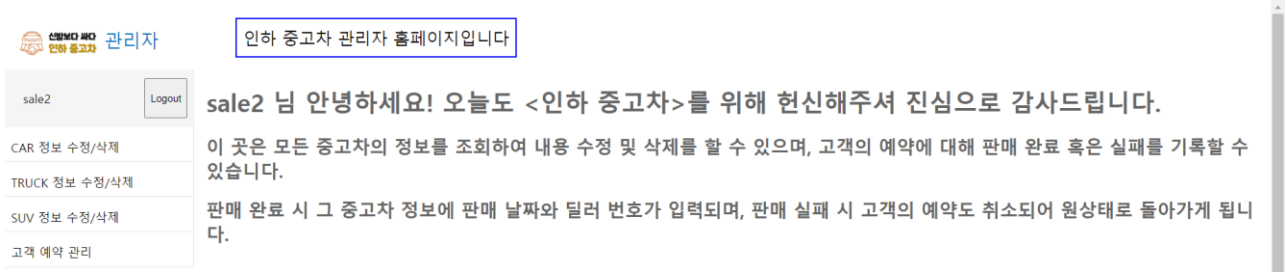
"CAR 구매 예약하기" 메뉴 클릭 -> 예약 가능한 CAR 조회 가능, car17 검색하여 513번 CAR 예약



"예약 조회/취소" 메뉴 클릭 -> 513번 CAR17이 예약자 cust2로 예약됨 확인



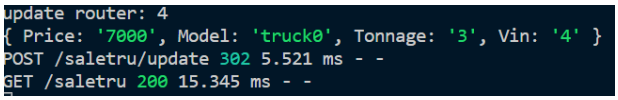
로그아웃 후 sale2(관리자)로 로그인, 비밀번호 2222



관리자 메인 페이지인 /salepage

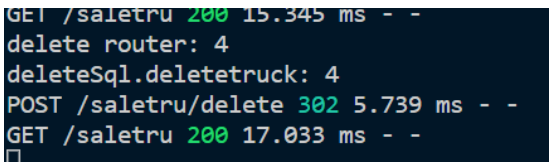
"CAR 정보 수정/삭제" 메뉴 클릭 -> 차량 정보 조회 및 검색 가능, 아까 예약한 정보도 확인 가능

"TRUCK 정보 수정/삭제" 메뉴 클릭 -> 4번 Truck 무게를 1에서 3으로 변경 후 수정 버튼 클릭



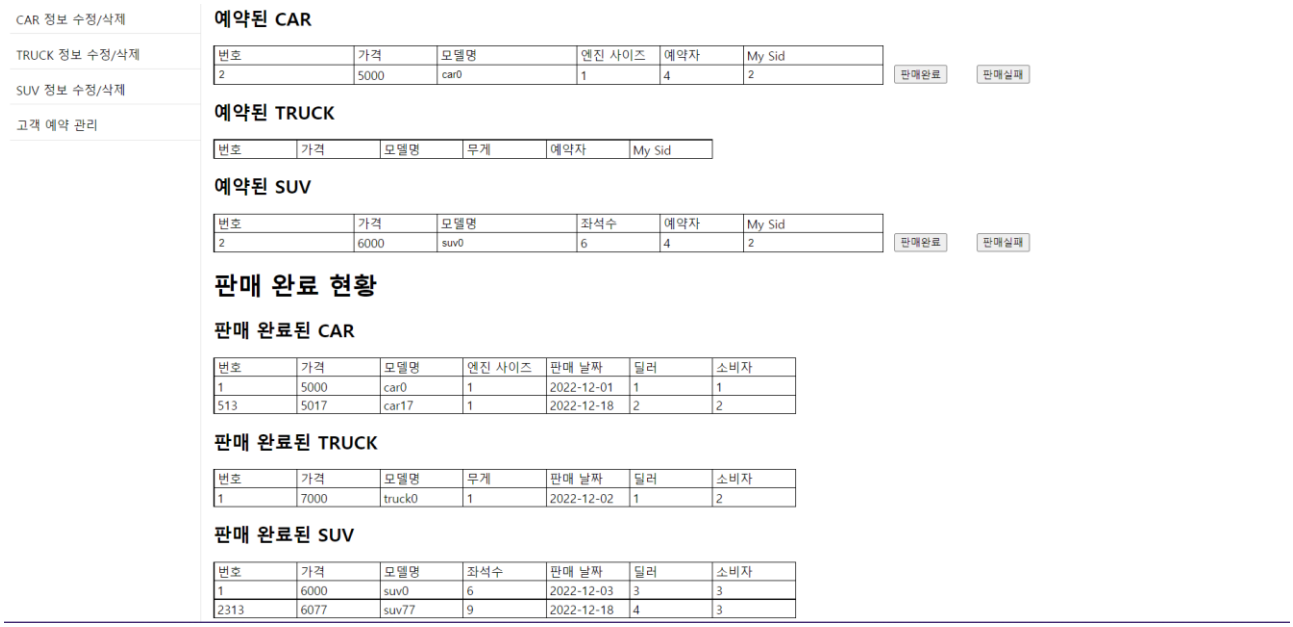
vscode 터미널에 수정된 데이터 출력 확인

4번 Truck 정보 삭제 위해 삭제 버튼 클릭

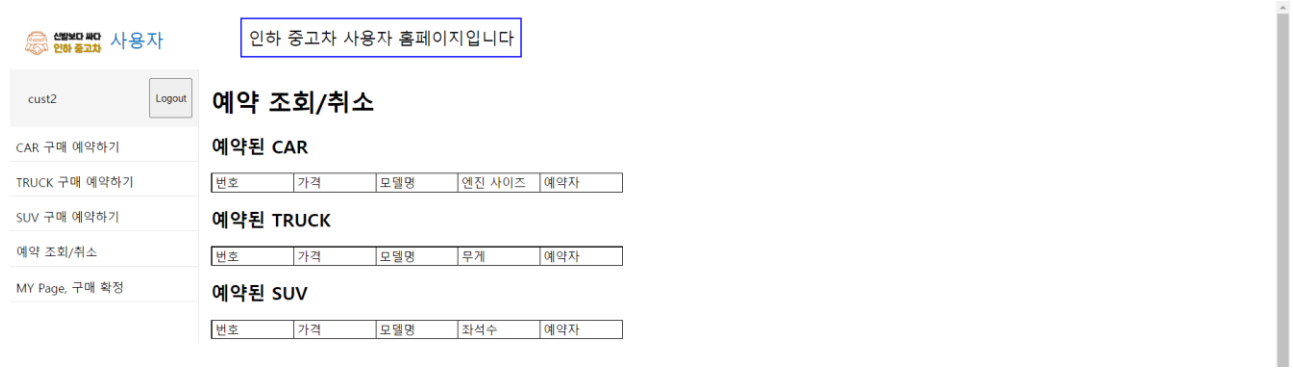


vscode 터미널에서 삭제된 데이터 출력 확인

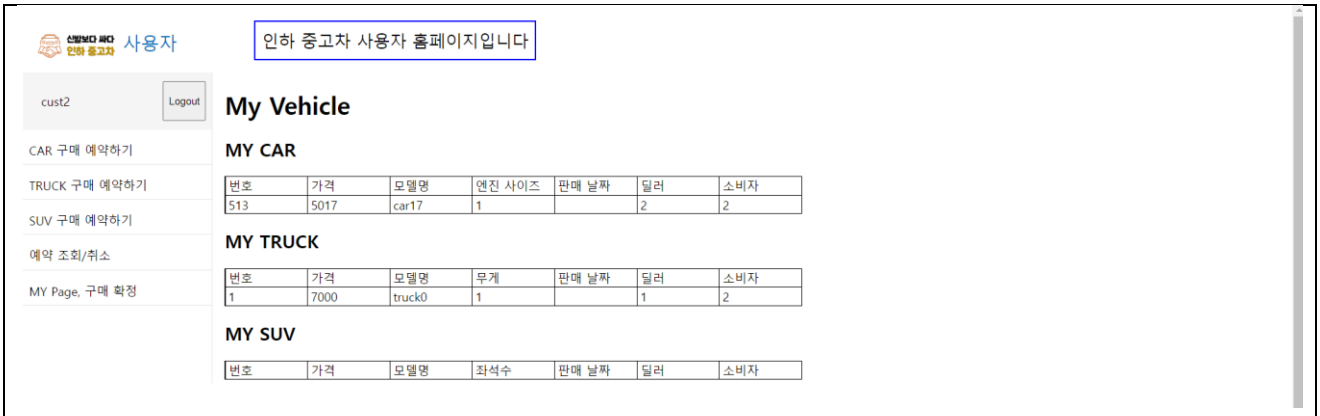
"고객 예약 관리" 메뉴 클릭 -> 모든 예약 및 판매 정보 조회 가능, 아까 예약한 513번 car17 정보 확인



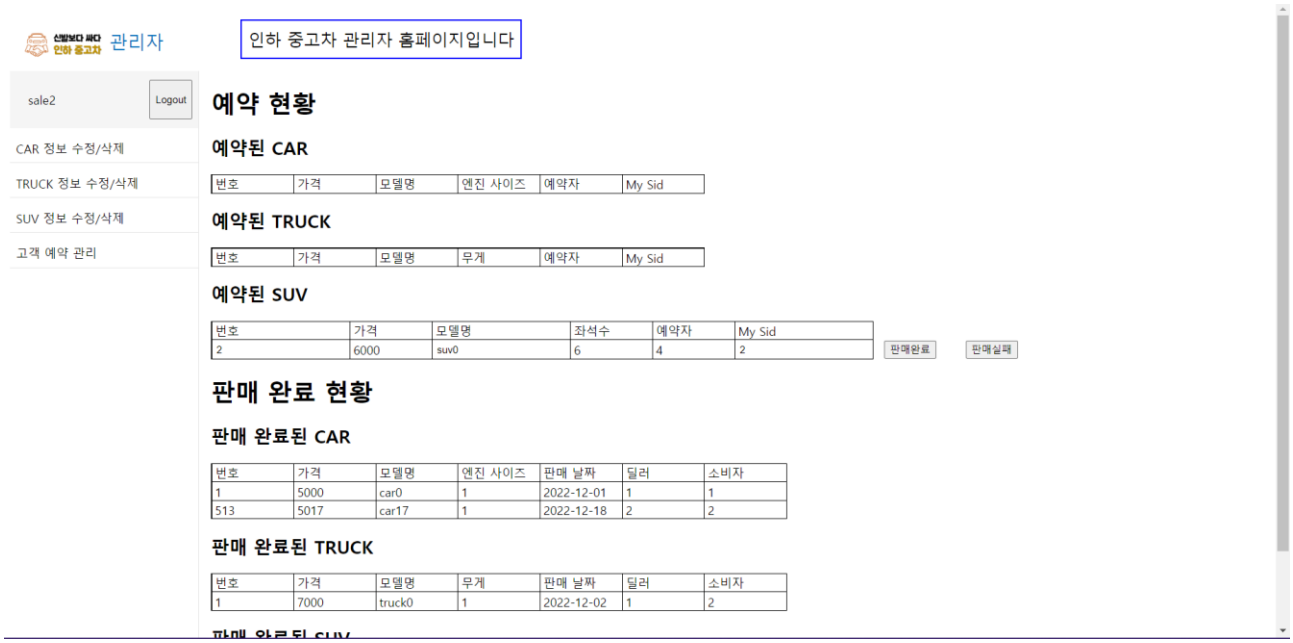
513번 car17 판매 완료 버튼 클릭 -> 오늘 날짜는 판매 날짜로, 로그인한 관리자가 딜러로 삽입 및 판매 완료 현황에 입력됨 확인

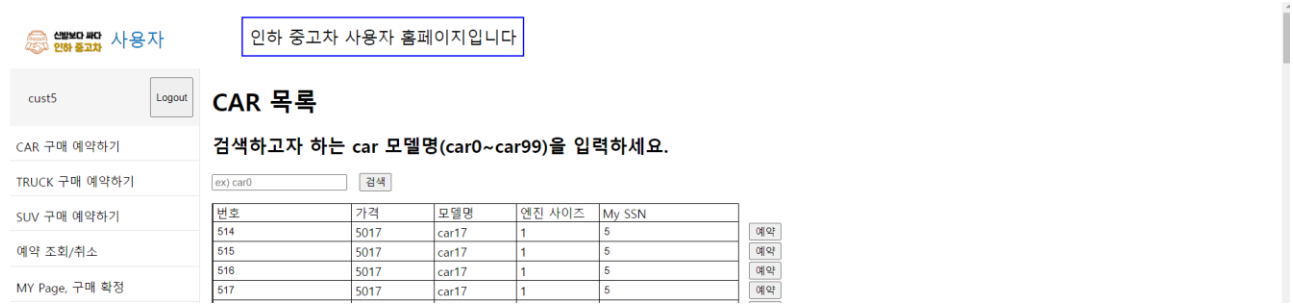


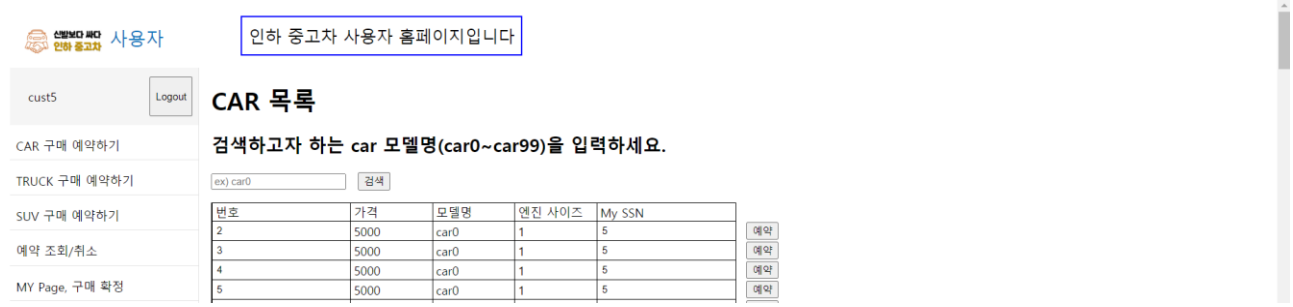
cust2로 다시 로그인하여 예약 조회 결과, 구매하였기 때문에 예약 조회에 나타나지 않음



"MY Page, 구매 확정" 메뉴 클릭 -> 구매 확정된 CAR 확인 가능







반대로 CAR에서 car0 판매 실패 클릭 -> 다른 사용자에서 car0 예약 가능 (513번 car17은 검색X)

Final Project			
ID	12201853	**Name**	김가현

결론

설계 프로젝트를 진행하면서 한 학기 실습 시간에 배운 내용을 총 정리할 수 있는 기회가 되었고, 내가 어떤 부분이 부족한 지도 깨닫게 되었다. 이번에 프로젝트를 구현하면서 가장 아쉬웠던 점은 10만 데이터를 색인 등을 이용하여 성능을 최적화하는 데 많이 부족했던 것이다. model로 index를 하기 위해 각 vehicle의 model 종류를 10개정도 잡았는데, model의 종류가 많아야지 index를 하는 것이 의미가 있다고 생각하여 model 종류를 350개로 하여 index를 진행하였다. 하지만 약 3만개의 vehicle을 출력하는 것은 무리가 있었다. 이를 해결하기 위해 검색 기능으로 모든 데이터를 볼 수는 없지만 일부를 볼 수 있도록 하였는데, 이를 통해 웹을 구현하는 것에 익숙해진 계기가 되었다.

또한 트랜잭션을 구현하는 것에 대해서는 실습 시간에 배우지 않아 개인적으로 공부하여 구현해보려고 했으나, 아직 웹 자체도 손에 완벽히 익지 않았기 때문에 과감히 수정해보지 못하였다. 따라서 설계 프로젝트를 제출하더라도 트랜잭션을 구현하여 사용자와 관리자 사이에서 데이터를 송신 및 수신할 때 충돌이 발생하더라도 rollback하여 재실행되도록 구현해보고 싶다.

또한 이번 프로젝트를 통해 버튼을 한 페이지에 두 개를 넣고, 자바스크립트에서 두개 이상의 post를 구현하는 것은 처음이라 버튼을 하나 눌러도 둘다 실행되는 등의 어려움을 겪었지만, 실행하는 url을 나누어 설계함으로써 원하는 결과를 내었다.