Database Project Report

<Yu-Gi-Oh! Tournament>

201720716 이재현 소프트웨어학과

201720812 전상우 소프트웨어학과

201820795 현수환 소프트웨어학과

201521495 양수훈 금융공학과

팀원 역할

이재현 : Front개발(카드 토너먼트), User승리횟수(Update) 구현, Delete

전상우 : Front 공동 개발 및 Debugging, 회원가입 구현, Delete, 크롤링

현수환 : DB제작(SQL문, Table) 및 Data Insert, Retrieval, 크롤링

양수훈 : Data 전처리, 문서 최종 검토 및 ppt제작, 문제점 및 수정사항 검토, 발표

공통 : 문서 작성 및 pack table 수작업

수정 사항

1차 문서에서 작성했던 테이블과는 다른 테이블들로 구성하였습니다. 우선 User의 정보가 들어있는 table을 추가하고, 각각의 카드 종류(몬스터, 마법, 함정)가 가지고 있는 테이블을 Type\_card, Type\_box, belong\_type 형식으로 분할하였습니다. type\_card table에는 카드 명과, 카드 속성 및 현재까지의 토너먼트 전적이 들어있고, type\_box에는 카드 명과 카드 효과가 들어있으며, type\_card에 카드 명을 외래키로 참조합니다. belong\_type table에는 카드의 serial number와 카드의 number, 카드 명, 카드 종류가 들어있으며, type\_card와 type\_box에 카드 명을 외래키로 참조하였고, serial number로 apack 테이블을 참조합니다. 카드들은 각각에 고유 pack에 소속되어 있기 때문에 pack table도 추가하였습니다.

저희가 기존에는 이상형 월드컵과 같은 형식으로 카드를 선택하는 서비스를 구현하고자 하였습니다. 그러기 위해서는 카드의 이미지도 크롤링을 하고, 그 이미지를 데이터베이스에 저장해야만 했습니다. 하지만 카드 수가 많아 이미지 저장시 토너먼트 실행이 오래 걸릴 것이고, 이미지를 크롤링 하기에는 부족한 부분이 있어서 이미지를 보여주는 형식에서, 카드의 특성들을 보여주는 형식으로 바꾸었습니다.

저희가 데이터를 긁어왔던 유희왕 데이터베이스는 몇몇 데이터들의 결함이 존재하였습니다. 저희는 pack table은 수작업으로 진행을 하였고, 몬스터 마법 함정카드들을 크롤링을 통해 데이터를 가져왔습니다. 하지만 pack에서 가져온 serial number와 크롤링으로 가져온 카드들의 serial number는 다른 부분들이 존재하였습니다. 그래서 기존에 sql파일을 가지고 데이터베이스에 추가하는 과정에서 오류가 발생하였습니다. 제약조건문은 문제가 없다고 판단하여 혹시 데이터에 결함이 있지 않나 확인하던 도중 결함의 존재를 알게 되었습니다. 그래서 저희는 그 부분들을 엑셀로 비교하여 하나씩 데이터 수정작업을 거쳤고, 그 후에는 오류없이 insert가 진행되었습니다.

Create Table

1. Account Table

| CREATE TABLE `account` (  `name` varchar(100) NOT NULL,  `ID` varchar(45) NOT NULL,  `password` varchar(45) NOT NULL,  `gender` varchar(45) NOT NULL,  `birth` date NOT NULL,  `email` varchar(45) NOT NULL,  check (gender in (`남’, `여’)),  PRIMARY KEY (`ID`),  UNIQUE KEY `email` (`email`)  ) ENGINE=InnoDB AUTO\_INCREMENT=71 DEFAULT CHARSET=utf8mb4; |
| --- |

2. Magic\_Card Table

| CREATE TABLE magic\_card (  card\_name varchar(100) NOT NULL,  icon varchar(10) DEFAULT NULL,  join\_count int NOT NULL DEFAULT 0,  vic\_count int NOT NULL DEFAULT 0,  win\_count int NOT NULL DEFAULT 0,  lose\_count int NOT NULL DEFAULT 0,  check(icon in('일반 마법', '속공 마법', '지속 마법', '필드 마법', '장착 마법', '의식 마법' )),  PRIMARY KEY (`card\_name`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci; |
| --- |

3. Magic\_box Table

| CREATE TABLE magic\_box (  card\_name varchar(100) NOT NULL,  card\_text text,  PRIMARY KEY (card\_name),  constraint connect\_magic foreign key(card\_name) references magic\_card(card\_name)ON DELETE CASCADE  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci; |
| --- |

4. Monster\_card Table

| CREATE TABLE monster\_card (  card\_name varchar(100) not null,  attribute varchar(10) NOT NULL,  constraint mon1 check(attribute in('어둠', '빛', '땅', '물', '화염', '바람', '신')),  card\_type varchar(10) NOT NULL,  constraint mon2 check(card\_type in ('마법사족', '드래곤족', '언데드족', '전사족', '야수전사족',  '야수족', '비행야수족', '악마족', '천사족', '곤충족', '공룡족', '파충류족', '어류족',  '해룡족', '사이킥족', '물족', '화염족', '번개족', '암석족', '식물족', '기계족',  '환신야수족', '환룡족', '사이버스족')),  grade\_size int NOT NULL,  type1 varchar(10) default NULL,  type2 varchar(10) default NULL,  constraint mon3 check(type1 in('일반', '효과',null)),  constraint mon4 check(type2 in('융합', '의식', '싱크로', '엑시즈', '링크',null)),  trait1 varchar(10) default NULL,  trait2 varchar(10) default NULL,  trait3 varchar(10) default NULL,  constraint mon5 check(trait1 in ('스피릿', '튜너', '듀얼',null)),  constraint mon6 check(trait2 in ('펜듈럼', '유니온', '튠',null)),  constraint mon7 check(trait3 in ('특수소환', '리버스',null)),  atk int NOT NULL ,  def int DEFAULT NULL,  constraint mon\_atk check(atk<=5000),  constraint mon\_def1 check(def<=5000),  constraint mon\_def2 check(def in(null)),  join\_count int NOT NULL DEFAULT 0,  vic\_count int NOT NULL DEFAULT 0,  win\_count int NOT NULL DEFAULT 0,  lose\_count int NOT NULL DEFAULT 0,  PRIMARY KEY (card\_name)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci; |
| --- |

5. Monster\_Box Table

| CREATE TABLE monster\_box (  card\_name\_box varchar(100) not null,  card\_text text,  PRIMARY KEY (card\_name\_box),  constraint connect\_monster  foreign key (card\_name\_box)  references monster\_card(card\_name)ON DELETE CASCADE  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci; |
| --- |

6. Trap\_Card Table

| CREATE TABLE trap\_card (  card\_name varchar(100) NOT NULL,  icon varchar(10) not NULL,  check(icon in ('일반 함정', '지속 함정', '카운터 함정')),  join\_count int NOT NULL DEFAULT 0,  vic\_count int NOT NULL DEFAULT 0,  win\_count int NOT NULL DEFAULT 0,  lose\_count int NOT NULL DEFAULT 0,  PRIMARY KEY (card\_name)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci; |
| --- |

7. Trap\_Box Table

| CREATE TABLE trap\_box (  card\_name VARCHAR(100) NOT NULL,  card\_text TEXT,  PRIMARY KEY (card\_name),  constraint connect\_trap foreign key (card\_name) references trap\_card(card\_name)ON DELETE CASCADE  ) ENGINE=INNODB DEFAULT CHARSET=UTF8MB4 COLLATE = UTF8MB4\_0900\_AI\_CI; |
| --- |

8. Pack Table

| CREATE TABLE `apack` (  `pack` varchar(100) NOT NULL,  `series` varchar(45) NOT NULL,  `release\_year` int not null,  `serial\_number` varchar(45) not null,  PRIMARY KEY (`pack`,`serial\_number`),  unique key (`serial\_number`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci; |
| --- |

9. Belong\_Monster Table

| CREATE TABLE belong\_monster (  serial\_number VARCHAR(45) not null,  card\_number int not null,  card\_name varchar(100) not null,  card\_type VARCHAR(45) NOT NULL,  check(card\_type in ('몬스터')),    constraint card\_name\_fk1 FOREIGN KEY(card\_name)REFERENCES monster\_card (card\_name) ON DELETE CASCADE,  constraint card\_name\_fk2 FOREIGN KEY(card\_name) REFERENCES monster\_box (card\_name\_box) ON DELETE CASCADE,    constraint serial\_number\_fk FOREIGN KEY(serial\_number)REFERENCES apack(serial\_number)ON DELETE CASCADE,  PRIMARY KEY (serial\_number, card\_number)  )ENGINE=INNODB DEFAULT CHARSET=UTF8MB4 COLLATE = UTF8MB4\_0900\_AI\_CI ; |
| --- |

10. Belong\_Magic Table

| CREATE TABLE belong\_magic (  serial\_number VARCHAR(45) not null,  card\_number int not null,  card\_name VARCHAR(100) NOT NULL,  card\_type VARCHAR(45) NOT NULL,  check(card\_type in ('마법')),  PRIMARY KEY (serial\_number, card\_number),  KEY card\_name\_idx (card\_name), key serial\_number\_idx(serial\_number),  constraint fk\_bm foreign key (serial\_number) references apack(serial\_number)ON DELETE CASCADE,  constraint card\_name3 foreign key(card\_name)references magic\_card(card\_name)ON DELETE CASCADE,  constraint card\_name4 foreign key(card\_name)references magic\_box(card\_name)ON DELETE CASCADE  ) ENGINE=INNODB DEFAULT CHARSET=UTF8MB4 COLLATE = UTF8MB4\_0900\_AI\_CI; |
| --- |

11. Belong\_Trap Table

| CREATE TABLE belong\_trap (  serial\_number VARCHAR(45) not null,  card\_number int not null,  card\_name VARCHAR(100) NOT NULL,  card\_type VARCHAR(45) NOT NULL,  check(card\_type in ('함정')),  PRIMARY KEY (serial\_number, card\_number),  KEY card\_name\_idx (card\_name),key serial\_number\_idx(serial\_number),  constraint fk6 foreign key (serial\_number) references apack(serial\_number)ON DELETE CASCADE,  constraint card\_name5 foreign key(card\_name)references trap\_card(card\_name)ON DELETE CASCADE,  constraint card\_name6 foreign key(card\_name)references trap\_box(card\_name)ON DELETE CASCADE  ) ENGINE=INNODB DEFAULT CHARSET=UTF8MB4 COLLATE = UTF8MB4\_0900\_AI\_CI; |
| --- |

Constraint

* Foreign Key

테이블의 가시성을 높이고, 무결성을 생각하여 각각의(몬스터, 마법, 함정) belong테이블들은 serial\_number를 apack의 serial\_number 참조하고, card\_name을 card와 box의 card\_name을 참조합니다. box는 card에서 파생 되기 때문에 box의 card\_name은 card의 card\_name을 참조합니다.

또한 card에서 데이터가 없으면 참조된 다른 테이블에서 또한 존재해선 안 되기에 on delete cascade로

자동으로 삭제되게하여 무결성을 지킵니다.

* Primary Key

모든 테이블은 primary key가 필요하기에 유일한 값을 가질 수 있는 attribute로 primary key를 구성하였습니다. type\_card, type\_box의 primary key는 카드 명으로 구성하였고, belong\_type 테이블은 카드의 고유 serial number를 primary key로 구성하였습니다. 그리고 User의 account 테이블은 User의 ID를 Primary key로 설정하였고, pack 테이블은 packl과 serial number로 primary key를 구성하였습니다.

* NOT NULL

카드 attribute들은 특성을 제외하고는 전부 값이 존재하기 때문에 Not Null 제약조건을 사용하였습니다. Not Null 조건이 들어가지 않은 속성들은, 카드마다 갖는 속성들이 다르기에 default로 Null로 설정후에, insert하면서 속성에 맞게 해당 attribute로 추가되는 형식으로 구성하였습니다.

* CHECK

카드의 속성 또는 종류는 정해져 있는 domain 내에서 결정되기 때문에, 결함 또는 오류로 인하여 domain 외에 속성에 대해 입력을 받지 않기 위해서 check 제약조건을 걸었습니다. 예시로 Trap\_card table에 있는 icon은 함정카드의 종류를 말하는 attribute로 일반, 지속, 카운터 함정으로 나뉩니다. 그렇기에 check(icon in ‘일반 함정', ‘지속 함정', ‘카운터 함정') 이라고 제약조건을 제시하여 그 외에 속성으로 입력, 수정이 안되도록 설정하였습니다.

Case Scenario

1. Insert

account table

|  |
| --- |

임의로 user의 이름을 고무-0 부터 고무-99999까지 추가하여, ID도 겹치지 않게 a1234부터 a11233까지 총 100,000개를 Insert하였습니다.

Monster\_card Table

|  |
| --- |

Monster\_box Table

|  |
| --- |

Magic\_card Table

|  |
| --- |

Magic\_box Table

|  |
| --- |

Trap\_card Table

|  |
| --- |

Trap\_box Table

|  |
| --- |

apack Table

|  |
| --- |

belong\_monster Table

|  |
| --- |

belong\_magic Table

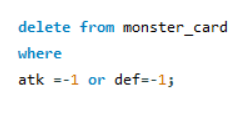
|  |
| --- |

belong\_trap Table

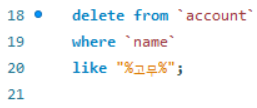
|  |
| --- |

기존에 크롤링으로 가져온 데이터를 CSV 파일로 저장 후에 .sql 파일로 수정하여 table에 추가하였습니다.

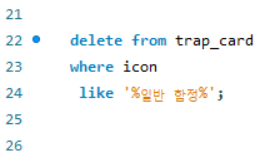
2. Delete



공격력 혹은 수비력이 -1(오류)인 카드를 삭제합니다.



이름에 ‘고무’가 있는 유저를 삭제합니다(매크로 유저)



일반 함정카드를 삭제합니다.

3. Update

승리 횟수 Count Up

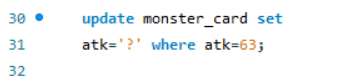
| query\_victory =  f'''UPDATE {session['card\_type']}\_card  SET win\_count=win\_count+1, join\_count=join\_count+1, vic\_count=vic\_count+1  WHERE card\_name="{winner[0]}"''' |
| --- |

참가 횟수 Count Up

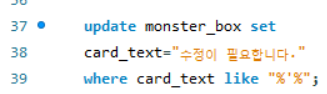
| query\_lose =  f'''UPDATE {session['card\_type']}\_card  SET lose\_count=lose\_count+1, join\_count=join\_count+1  WHERE card\_name="{loser[0]}"''' |
| --- |

회원정보 수정

| sql =  "UPDATE account SET name = %s, password = %s, gender = %s, birth = %s, email = %s WHERE ID = %s" |
| --- |

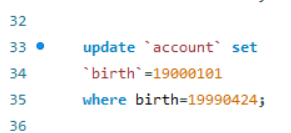


Atk가 ‘?’일 경우 63(ASCII)로 업데이트 합니다.



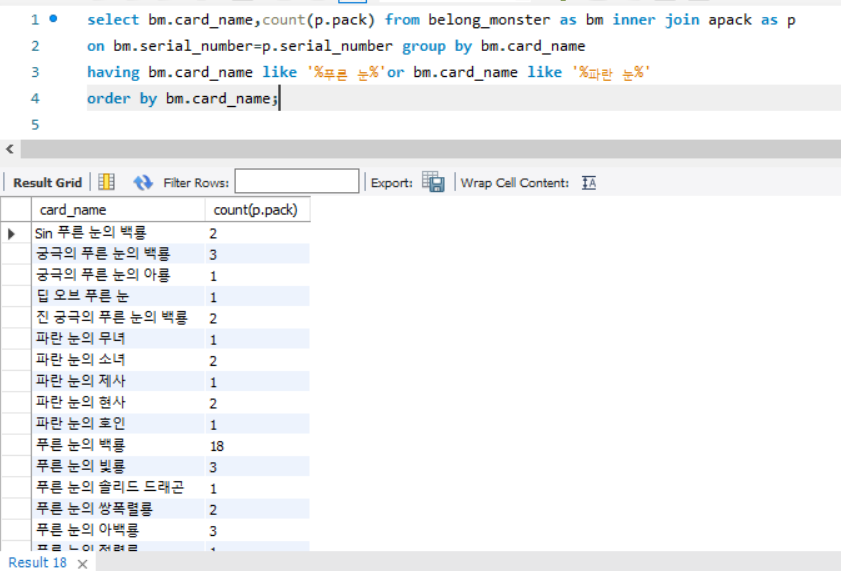
monster\_box에서 card\_text가 실제 db와 다른 text를 바꿉니다.

(실제 text는 “ ‘ “가 아닌 ‘ “ ‘가 쓰입니다.)

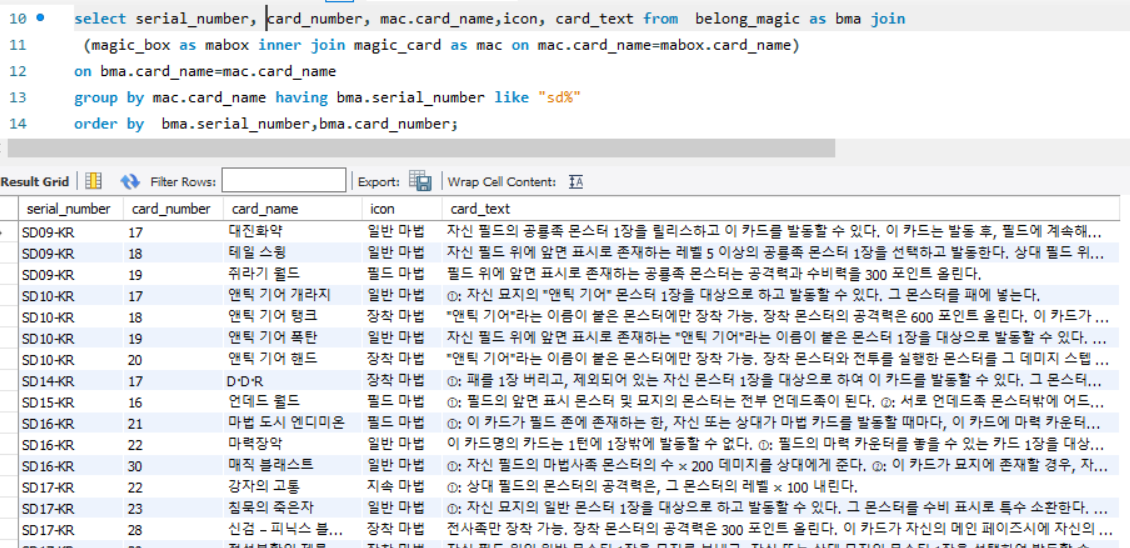


account에서 생년월일을 임의로 변경합니다.

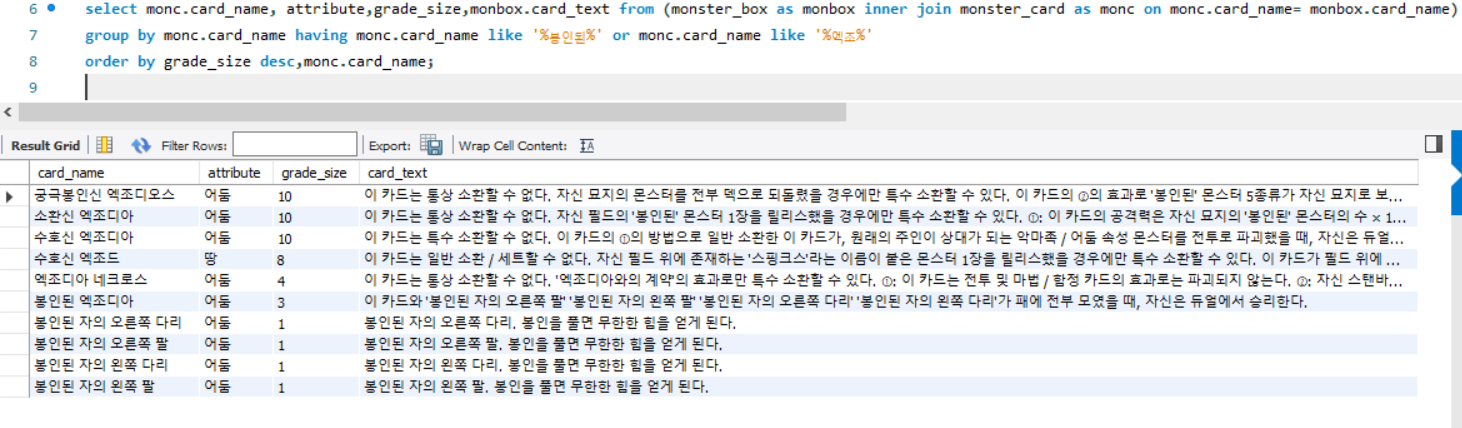
4. Retrieval



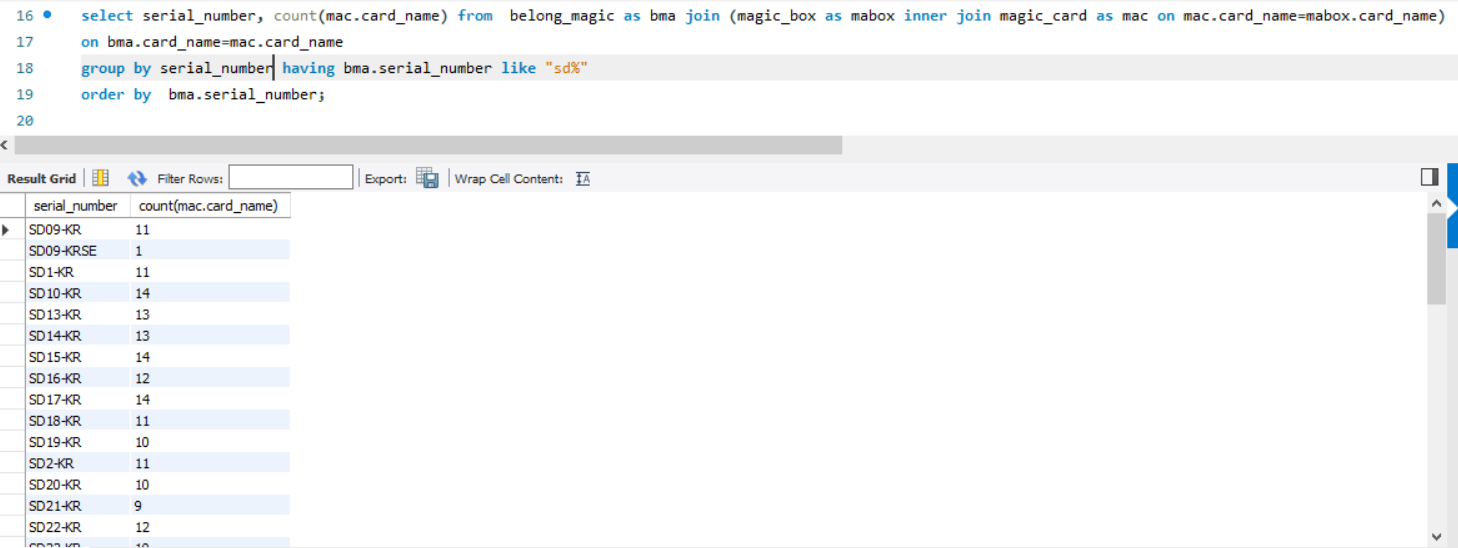
card\_name에 ‘푸른 눈’과 ‘파란 눈’이 포함된 몬스터 카드와 수록 횟수를 검색했습니다.



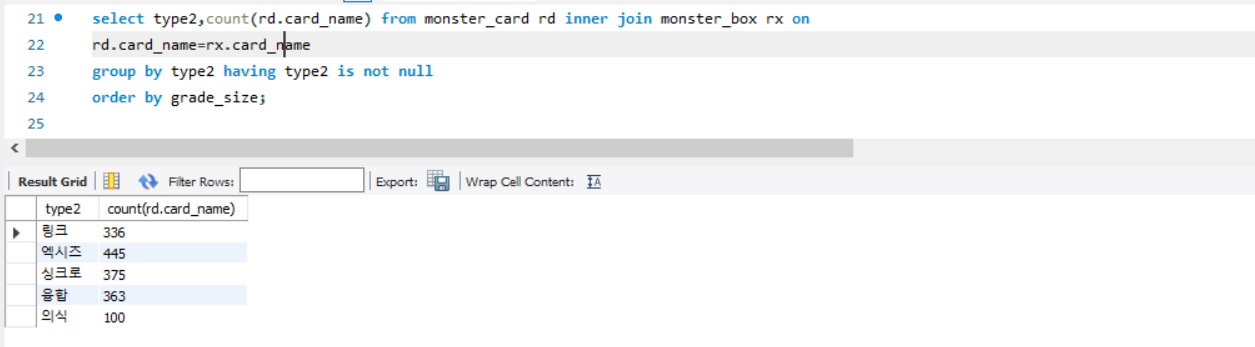
serial\_number에 ‘SD’를 포함하는 모든 마법카드를 카드효과를 포함하여 검색했습니다.



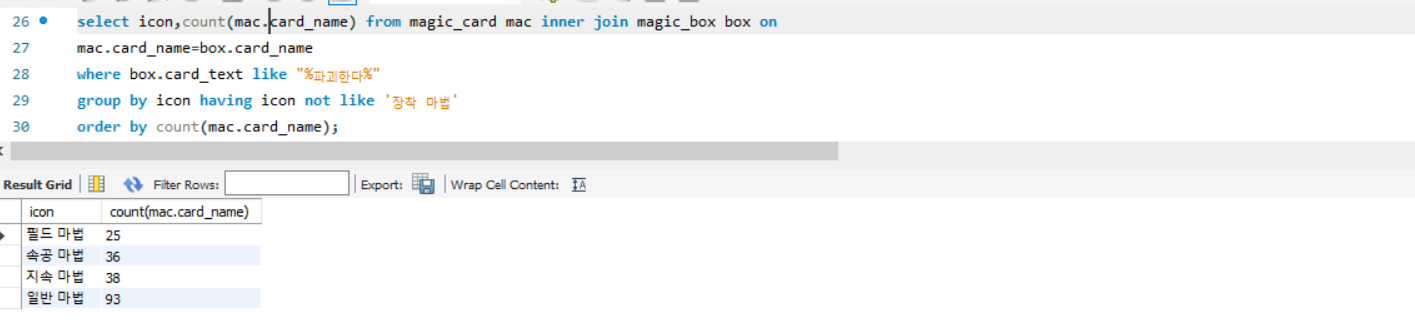
card\_name에 ‘봉인된’이나 ‘엑조’가 포함된 모든 몬스터카드를 card\_text를 포함하여 검색했습니다.



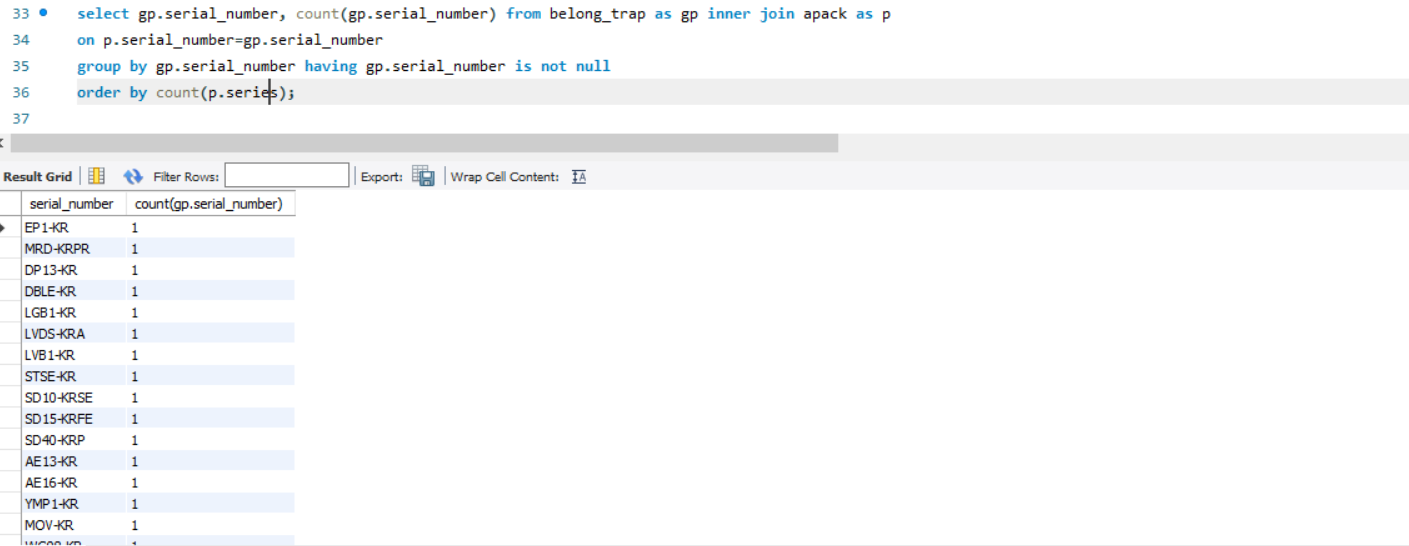
serial\_number에 ‘sd’를 포함하는 카드 중 마법 카드의 개수를 serial\_number에 따라 그룹화해서 검색해보았습니다.



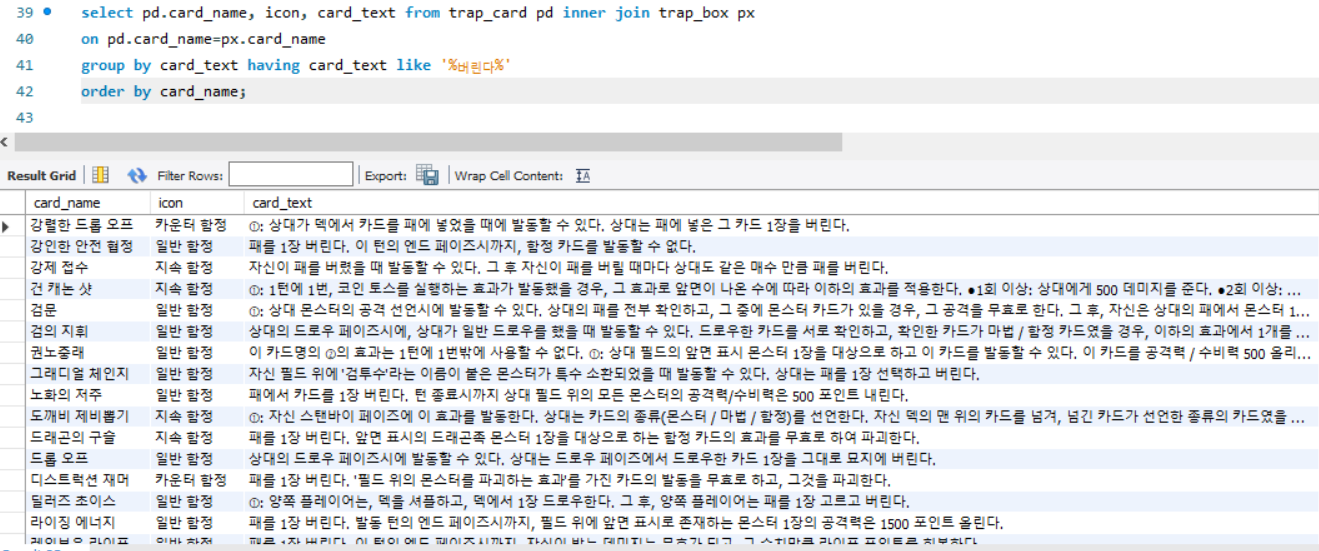
monster\_card의 type2가 null이 아닌 카드들을 type2로 그룹화 시켜 검색해보았습니다.



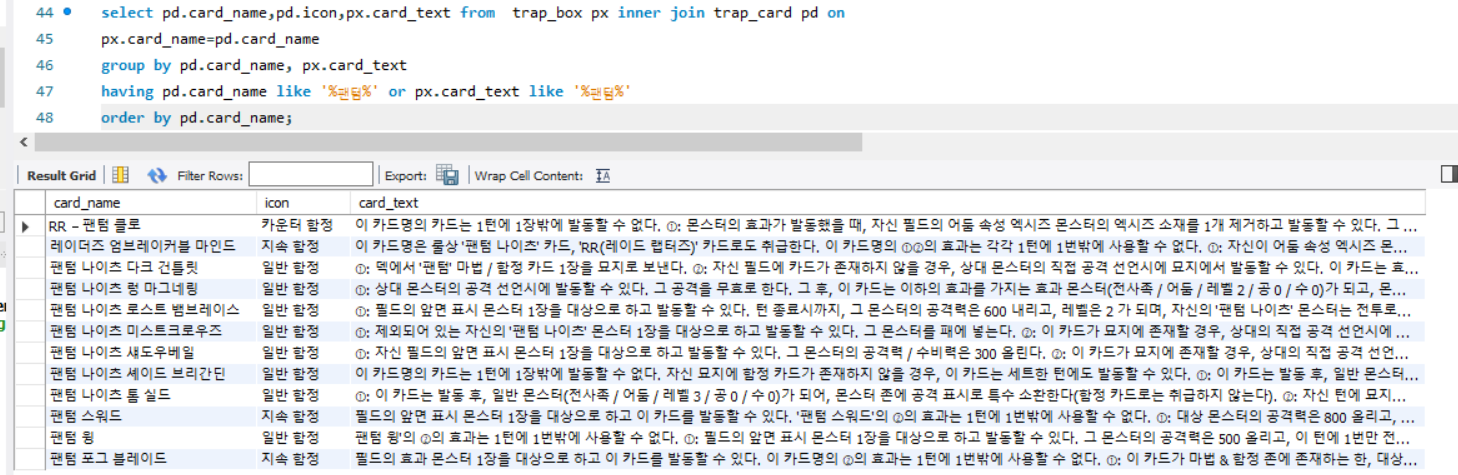
마법카드 중 icon이 ‘장착마법’이 아닌 카드 중에서 text에 ‘파괴한다’가 포함된 카드들을 icon으로 그룹화하여 검색했습니다.



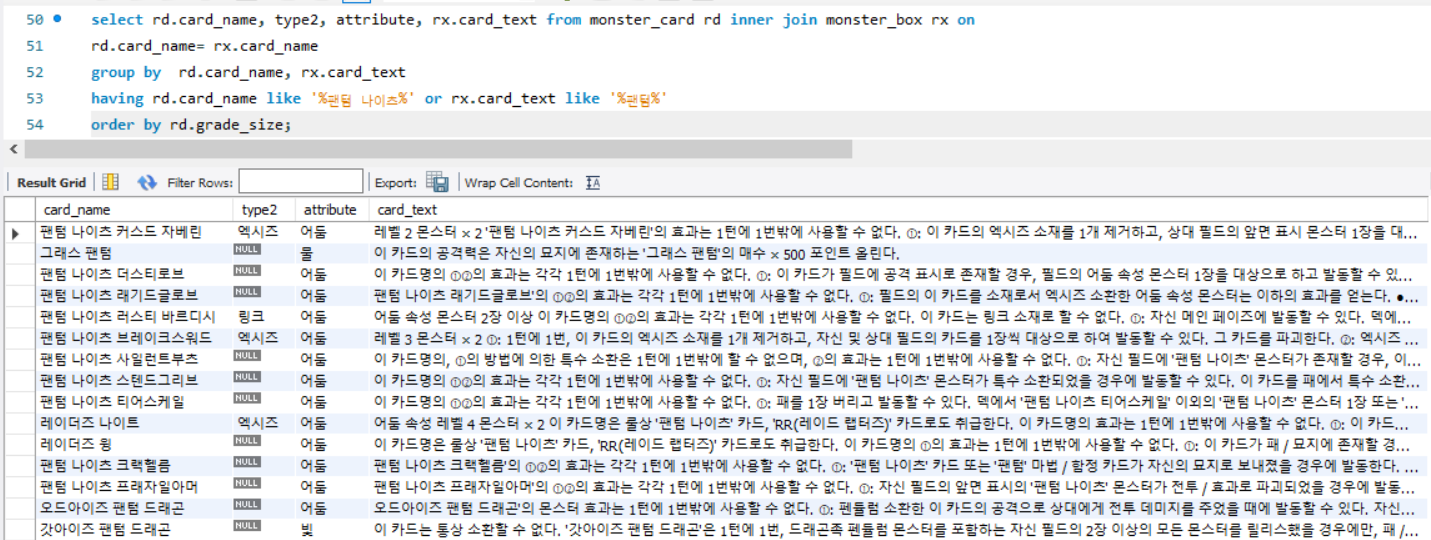
각 serial\_number에 몇장의 함정카드가 있는지 검색해보았습니다.



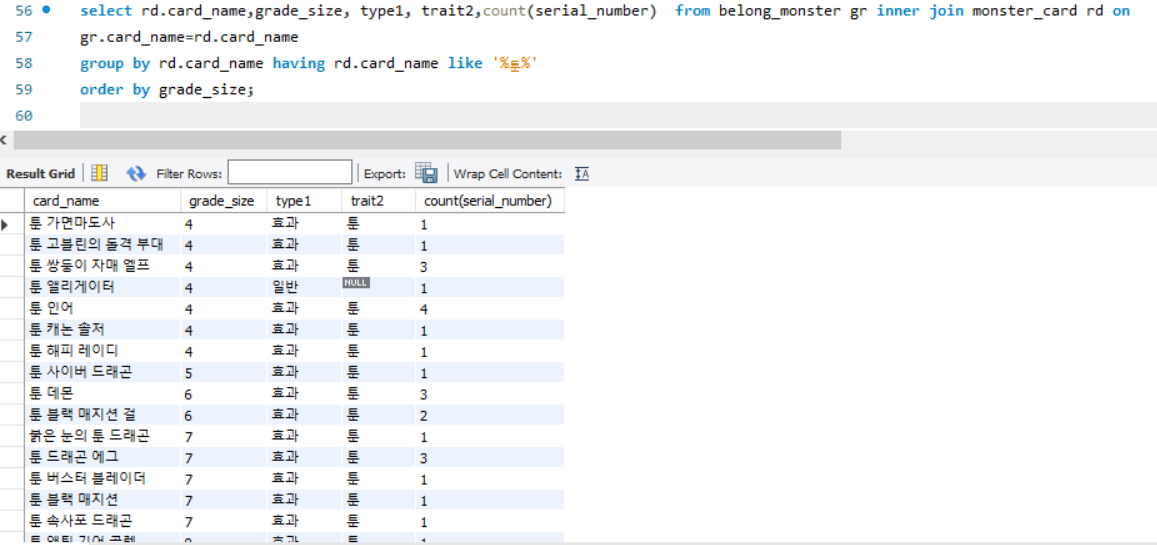
함정 카드 중 card\_text에 ‘버린다’가 포함된 카드들을 검색해보았습니다.



함정 카드 중 card\_name 또는 card\_text에 ‘팬텀’이 포함된 카드만을 검색해보았습니다.



몬스터 카드 중 card\_name에 ‘팬텀 나이츠’나 card\_text에 ‘팬텀’이 기재된 카드만을 검색해보았습니다.



카드 이름에 ‘툰’이 있는 몬스터 카드을 grade\_size로 정렬시켜 검색해보았습니다.

Result

검색 사이트에서 구현한 기능들을 직접 동작시켜 보겠습니다.



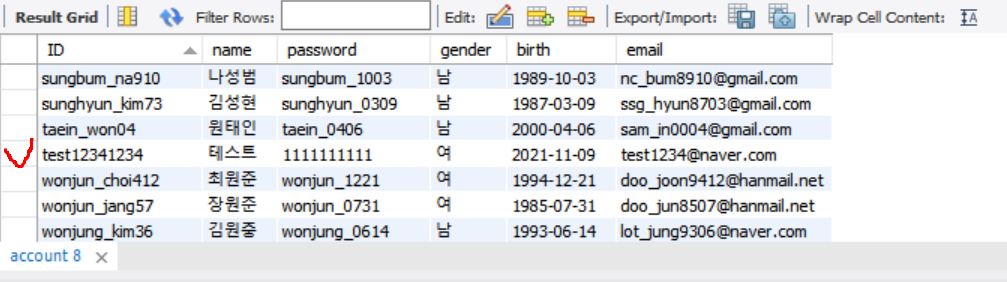
위 사진은 검색 사이트의 메인 페이지입니다.

속성들을 원하는 것들만, 레벨이나 공격력 등은 원하는 범위로 설정하여 검색할 수 있습니다.

우측 상단은 로그인 및 회원가입 기능을 구현해 놓은 것입니다.



우선 우측 상단의 회원가입을 클릭해 위와 같이 정보를 기입한 후 가입 버튼을 눌러보았습니다.



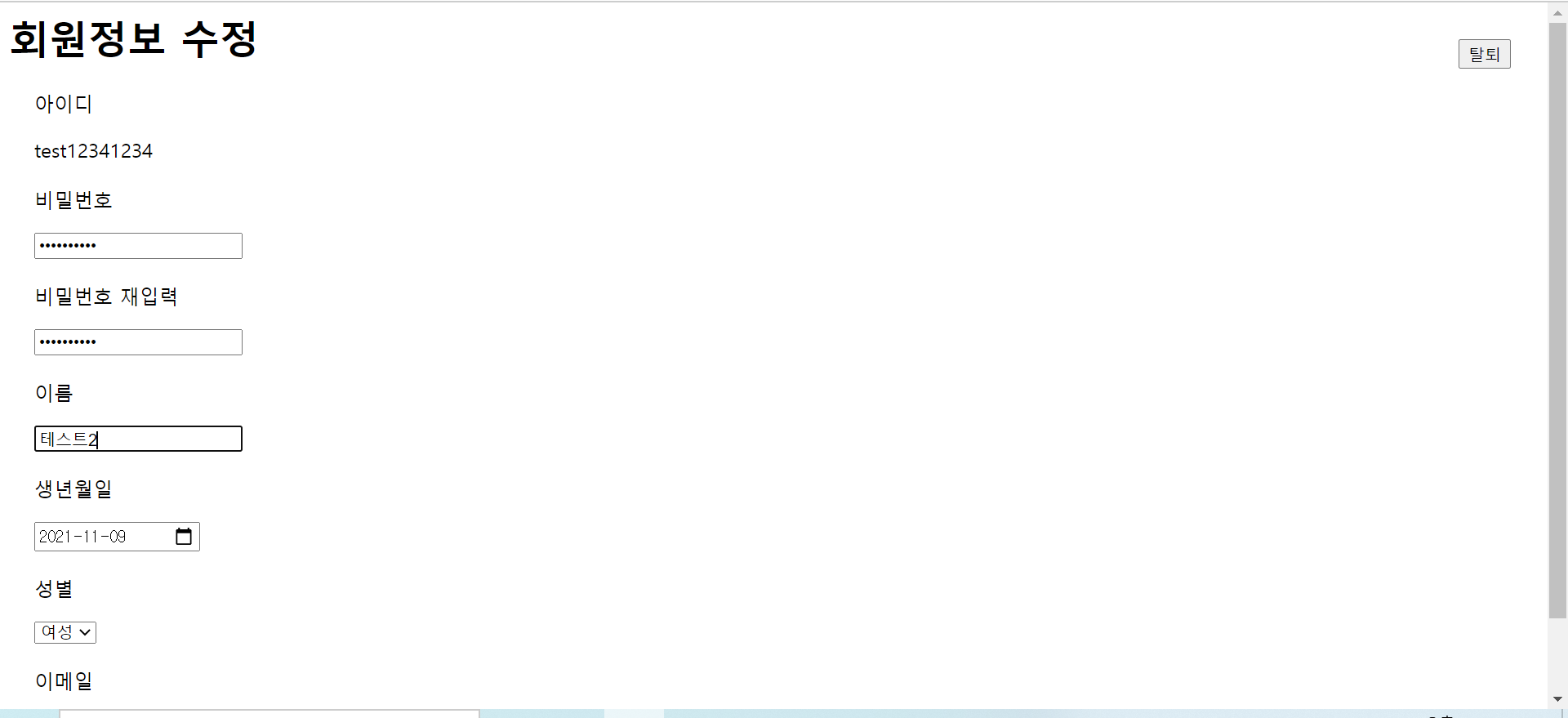
연결된 database의 account 테이블에서 확인해보니 방금 가입한 사용자의 정보가 추가됐음을 확인할 수 있습니다.



가입한 아이디와 비밀번호로 로그인을 해보겠습니다.



로그인 결과 화면입니다. 우측 상단의 로그인 및 회원가입란이 사라졌고 좌측 상단의 유저 아이디, 로그아웃, 회원 정보 수정 기능이 생성됐음을 확인할 수 있습니다.



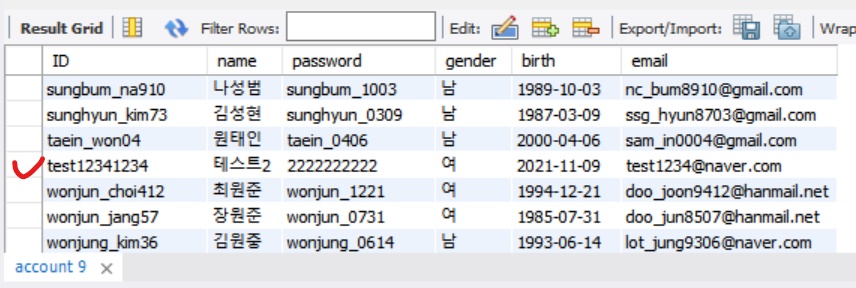
다음으로 회원 정보를 수정해보겠습니다.

아이디는 수정하면 안되므로 텍스트 형식으로 나타내 수정할 수 없도록 하였고

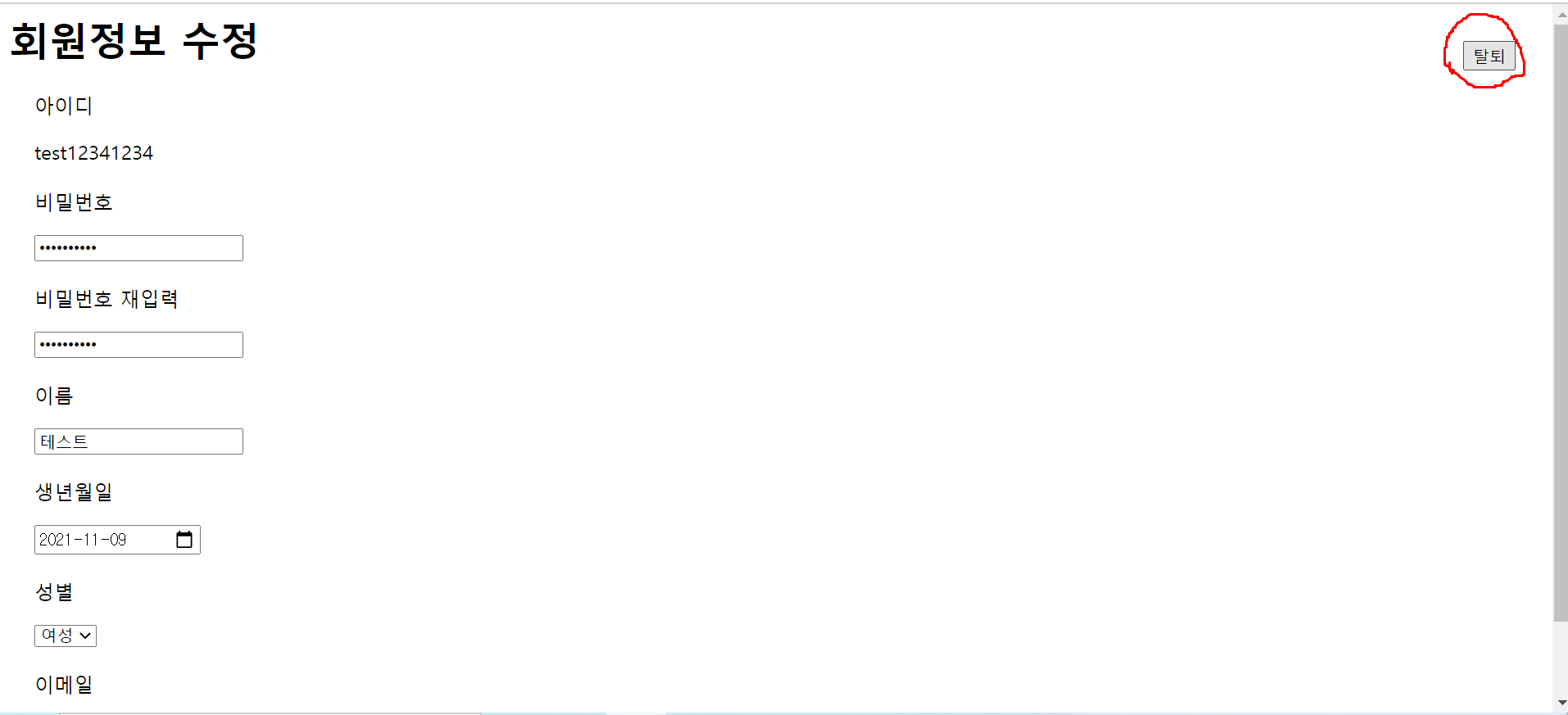
나머지 정보는 직전의 회원 정보를 default값으로 나타내게 하여 수정을 원하지 않는 부분은

건드리지 않고 제출할 수 있도록 했습니다.

이름, 비밀번호를 각각 테스트2, 2222222222로 수정해보겠습니다.



수정 뒤 database의 account 테이블을 확인해보니 name과 pasword가 수정되었음을 확인할 수 있습니다.



다음으로 탈퇴 기능을 사용해보겠습니다.

회원 정보 수정에서 우측 상단의 탈퇴 버튼을 눌러 회원 탈퇴를 진행한 뒤 database를 확인해 보겠습니다.



테스트2를 name으로 가지는 row가 삭제되었음을 확인할 수 있습니다.



다음으로 검색 기능을 직접 사용해보겠습니다.

위와 같은 조건으로 설정하면

카드 이름엔 ‘푸른 눈의’를 포함한, 레벨은 5이상 12이하(최대 레벨 미설정시 최대 레벨인 12로 설정),

공격력은 1000이상 5000이하(최대 공격력 미설정시 최대 공격력인 5000으로 설정),

수비력은 1000이상 5000이하(최대 공격력 미설정시 최대 수비력인 5000으로 설정),

속성은 어둠, 빛, 물, 화염 중에 하나,

종족은 마법사족, 드래곤족, 야수족, 악마족, 번개족, 식물족, 사이킥족, 사이버스족 중 하나인

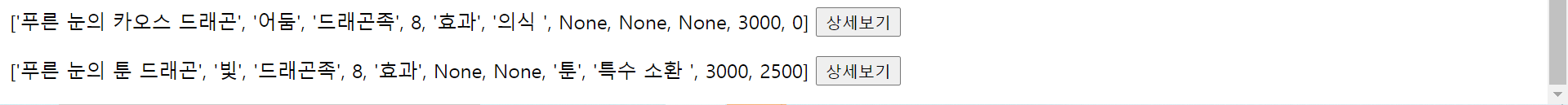
몬스터 카드를 검색하게 됩니다.

우선 실제 유희왕 카드 db 사이트에서 위와 같은 조건으로 검색한 후 검색 결과가 일치하는지 확인해보겠습니다.





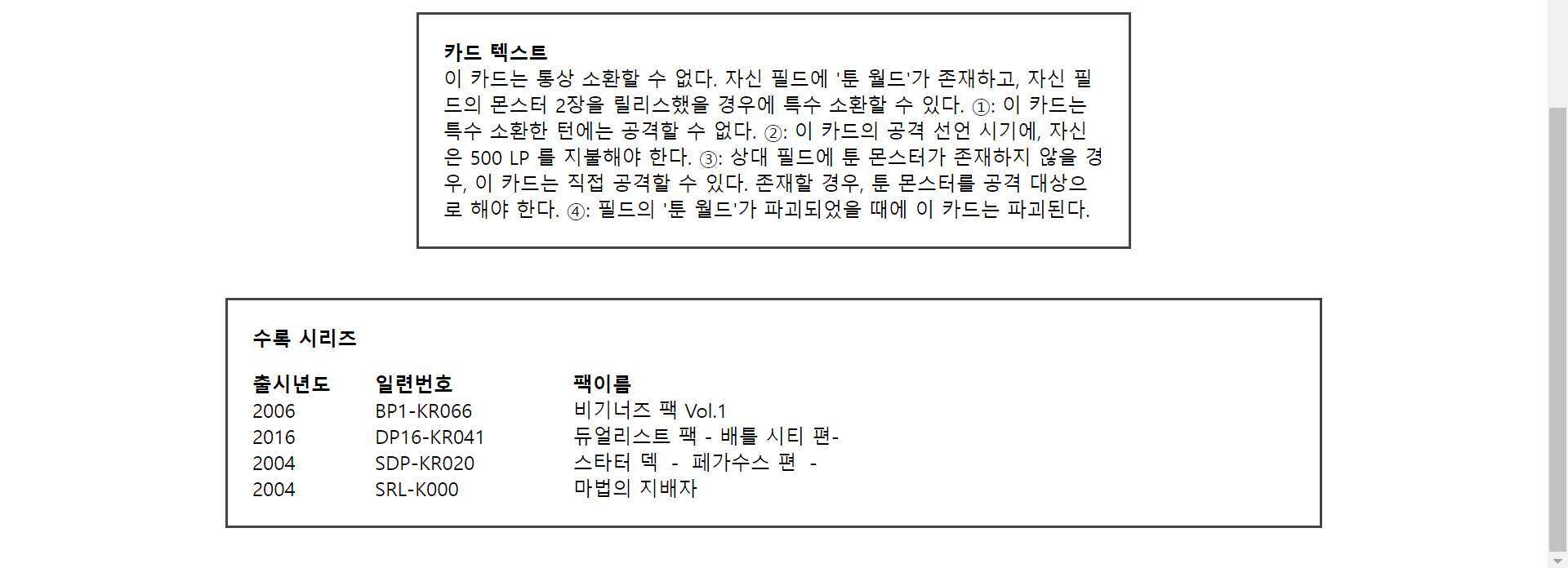




위의 두 결과 사진을 보면 각각 13개로 검색 결과가 일치함을 확인할 수 있습니다.

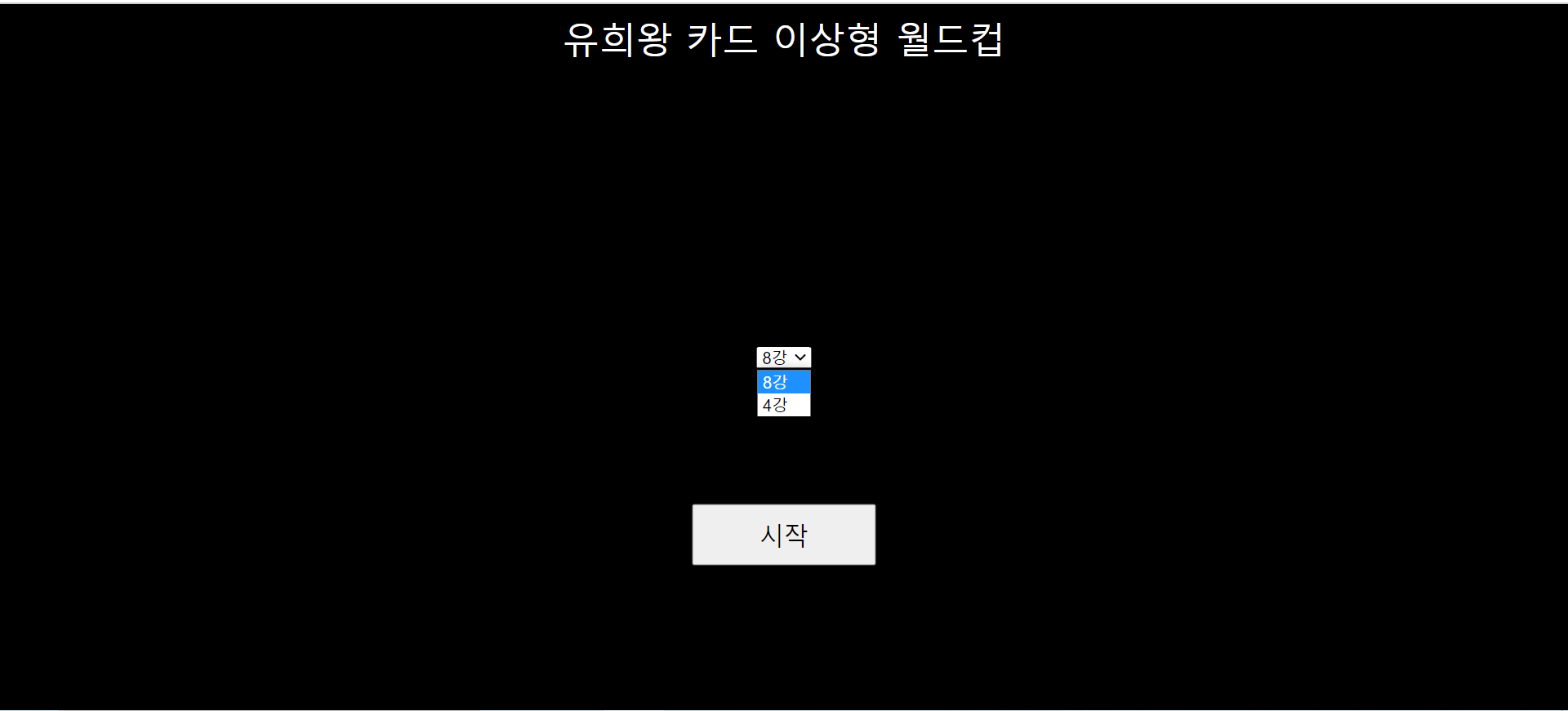
여기서 ‘푸른 눈의 툰 드래곤’의 정보를 확인하기 위해 상세보기 버튼을 클릭해 보겠습니다.





상세보기 버튼을 클릭하면 위와 같이 카드에 대한 전체 정보를 확인할 수 있습니다.

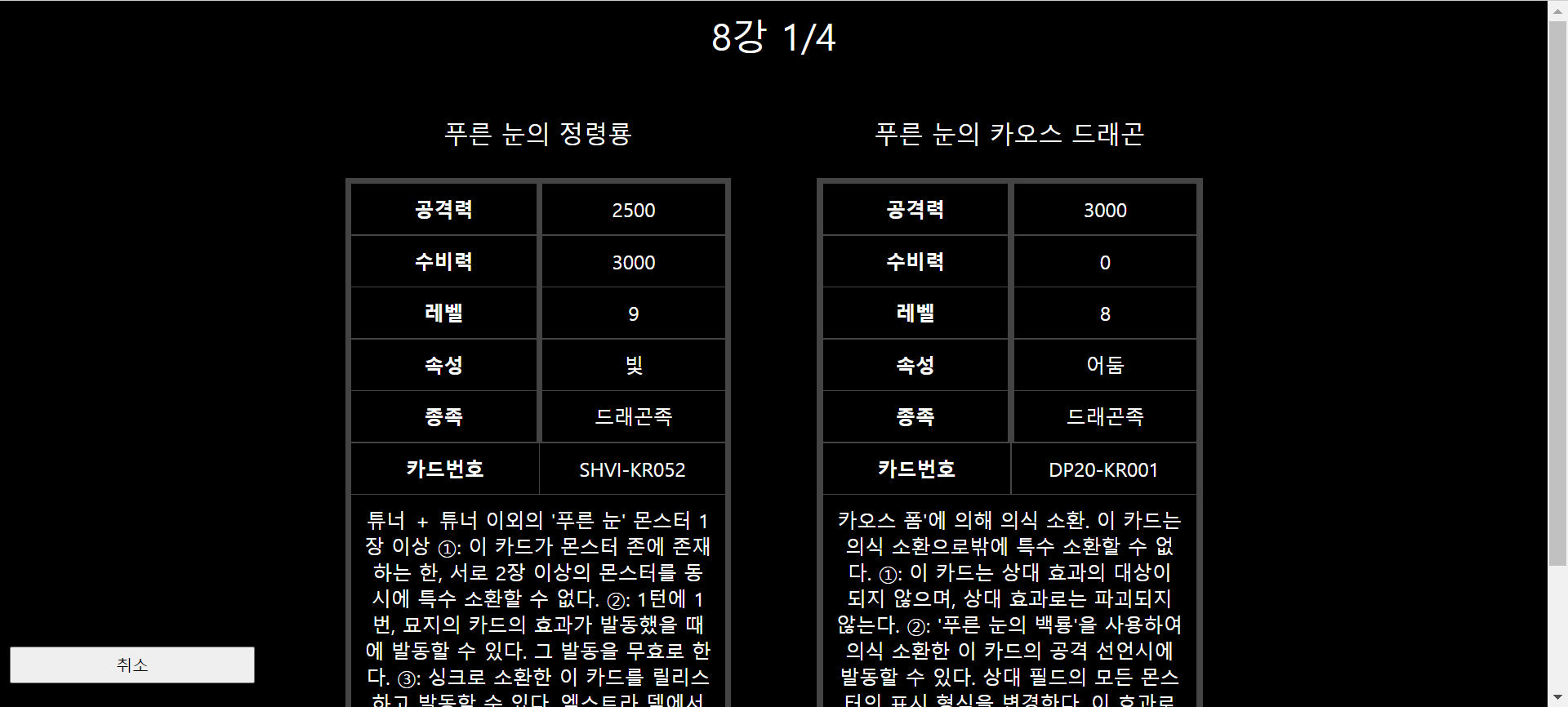
이제 뒤로가기 버튼을 눌러 직전 페이지로 이동하고 거기서 월드컵시작 버튼을 눌러 이상형 월드컵을 진행해 보겠습니다.



위는 이상형 월드컵 첫 페이지 입니다. 여기서 원하는 라운드 수(O강)을 선택할 수 있습니다.

라운드 수를 선택하면 전체 검색 결과 카드에서 shuffle한 뒤 라운드 수 만큼의 카드를 꺼내 이상형 월드컵을 진행하게 됩니다.

최대 64강까지 진행할 수 있게 했고 카드 수를 고려하여 카드 수보다 많은 라운드는 선택할 수 없게 하였습니다. 사진의 경우 검색 결과가 13장인 카드들에 대하여 이상형 월드컵을 진행하므로 16강부터는 선택할 수 없게 설정하였습니다. 8강을 선택하여 월드컵을 진행해 보겠습니다.



위는 월드컵 중 일부 장면입니다. 좌우에 각각의 카드에 대한 정보가 나와있어 서로 비교할 수 있고 원하는 카드정보를 나타낸 표를 클릭하면 해당 카드는 다음 라운드로 진출하게 됩니다.

각 라운드가 끝날 때마다 해당 라운드의 참가 카드에 대한 참가 횟수, 승리 횟수, 패배 횟수가 갱신되고

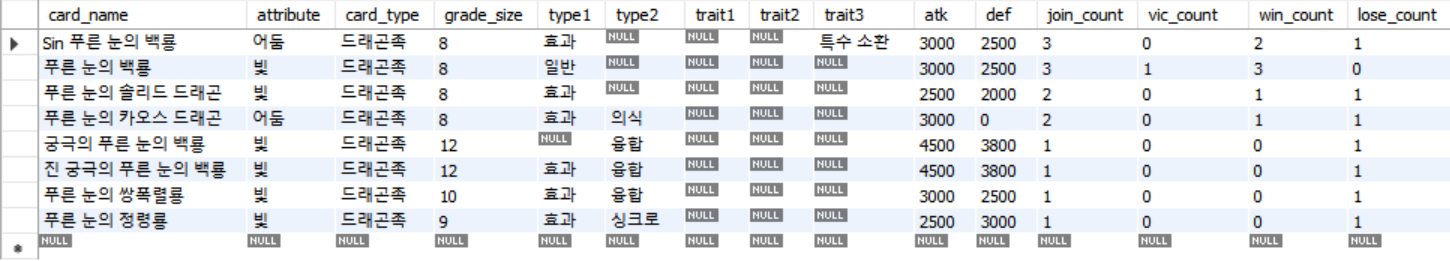
결승전의 경우 추가로 우승 횟수도 갱신됩니다.



위 사진은 월드컵이 끝난 후 우승 카드를 나타내는 화면입니다.

다시하기 버튼을 누르면 월드컵 첫 페이지로 돌아가 전체 검색 결과인 13개의 카드에 대하여 새로운 월드컵을 진행할 수 있습니다.

처음화면 버튼을 누르면 검색 메인페이지로 돌아가 새로운 검색을 할 수 있습니다.



위는 월드컵(8강)에 참여한 8장의 카드를 나타낸 monster\_card 테이블의 일부분 입니다.

8강의 경우

8강에서 패배한 카드는 8강만 참가했고 8강서 패배했으므로

join\_count=1, win\_count=0, lost\_count=1

4강에서 패배한 카드는 8강, 4강에 참여했고 8강서 승리, 4강서 패배했으므로

join\_count=2, win\_count=1, lost\_count=1

결승전을 진행한 두 카드는 8강, 4강, 결승에 참여했으므로 join\_count=3이고

우승 카드는 8강, 4강, 결승에서 승리하였으므로 win\_count=3, vic\_count=1, lose\_count=0

패배 카드는 8강, 4강서 승리, 결승서 패배했으므로 win\_count=2, lose\_count=1 이어야하는데

위의 테이블의 join\_count, vic\_count, win\_count, lose\_count과 비교해보면 이와 일치함을 확인할 수 있습니다.