|  |
| --- |
| **Project 2** |

|  |  |  |  |
| --- | --- | --- | --- |
| **2024년 2학기** | | | |
| 제출일 | | 2024. 11. 02. | |
| 과목명 | | 데이터베이스및데이터시각화 | |
| 담당교수 | | 이기훈 | |
|  | | | |
| 학과 | 컴퓨터정보공학부 | 학과 | 컴퓨터정보공학부 |
| 학번 | 2020202034 | 학번 | 2020202033 |
| 이름 | 김태완 | 이름 | 고동우 |

|  |  |  |  |
| --- | --- | --- | --- |
| 학과 | 컴퓨터정보공학부 | 학과 | 컴퓨터정보공학부 |
| 학번 | 2020202031 | 학번 | 2020202054 |
| 이름 | 김재현 | 이름 | 이종혁 |



**1.웹 애플리케이션 소개**

이번 데이터베이스및시각화의 프로젝트 주제는 사람들이 물건을 사고파는 중고 장터를 직접 개발하는 것이다. 중고 장터는 다양한 연령대의 사람들이 다양한 종류의 상품을 거래하는 웹 사이트이다. 여기서 많은 회원의 데이터나 물건의 데이터가 발생하게 된다. 이러한 데이터들을 관리하고 적절하게 사용하여 회원이 물건을 판매하거나 원하는 물건을 편리하고 간편하게 거래할 수 있게 도움을 주는 중고 장터를 개발한다.

개발할 중고 장터는 회원과 관리자가 사용한다. 회원은 물건을 판매하거나 구매한다. 회원은 구매를 희망하는 물건을 검색할 수 있다. 이때 물건의 카테고리(태그), 가격, 거래 방식, 거래 지역을 필터를 통해서 상세하게 검색할 수 있다. 검색된 물건은 물건의 정보뿐만 아니라 판매자의 정보 또한 볼 수 있다. 판매자가 판매한 물건이나 후기, 신뢰도, 별점을 통해서 믿을만한 판매자인지 확인할 수 있다. 판매 물건에 대해 궁금한 것이 있을 때는 판매자에게 문의를 통해 질문을 할 수 있다. 구매가 완료된 물건은 구매자가 후기를 작성할 수 있다. 부적절한 물건에 대해서는 신고도 할 수 있다. 시세조회에서는 다양한 물건들을 카테고리, 연령대 등을 기준으로 데이터로 시각화된 정보를 볼 수 있다. 회원은 마이페이지에서 회원정보수정, 나의 판매/구매 내역, 찜한 상품 정보를 확인할 수 있다. 회원은 자유게시판에서 게시글을 작성하여 다른 회원들과 소통도 가능하다. 관리자는 회원의 정보를 볼 수 있으며, 신고가 들어왔을 때 신고된 물건이나 회원에 대해서 적절한 조치를 취할 수 있다. 그리고 자유게시판을 통해서 공지글을 작성할 수 있다.

**2.구현 환경 기술**

웹 프론트엔드에서 Vue.js를 사용하여 사용자 인터페이스를 개발한다. 데이터가 실시간으로 변경될 경우 Vue.js가 화면을 자동으로 업데이트하여 사용자에게 최신 정보를 제공한다. 따라서 Vue.js의 반응형 데이터 바인딩은 사용자 및 거래 데이터를 시각화하는 데 매우 유용하다. 또한 Vue.js는 간결하고 직관적인 문법을 갖추고 있어 초보자도 쉽게 학습할 수 있다. 따라서, 다른 프론트엔드 프레임워크에 비해 빠르게 익숙해질 수 있다.

웹 개발에서 널리 사용되는 JavaScript로 프론트엔드와 백엔드가 동일한 언어를 사용하여 개발한다. 이렇게 하면 언어 전환에 대한 부담이 줄어든다. Node.js 환경에서 Express 프레임워크를 사용하여 서버를 구축한다. Node.js와 Express는 이전에 사용해본 경험이 있는 기술로, 익숙한 환경에서 서버를 구축하는 것이 개발 효율성을 높일 수 있다고 판단했다. Express는 가벼운 프레임워크로서 RESTful API 설계와 같은 HTTP 요청 처리에 최적화되어 있어 빠르고 효율적으로 서버를 구성할 수 있다.

중고 거래 플랫폼의 제품 정보, 사용자 정보, 거래 내역 등을 저장하기 위해 관계형 데이터베이스인 MySQL을 사용한다. MySQL은 데이터 구조가 명확하고 안정성이 높아 중고 장터 애플리케이션의 중요한 데이터 저장에 적합합니다. 또한, 이미 MySQL을 사용해 본 경험이 있기 때문에 개발을 더욱 수월하게 진행할 수 있다.

**3.모든 기능 설명**

* 기본 기능

**이용자 기능**

중고 장터 회원은 회원가입/로그인을 통해 중고 장터를 이용할 수 있다. 상품 조회를 통해서 회원은 원하는 상품을 검색할 수 있다. 검색한 상품에 궁금한 것을 상품 판매자에게 문의할 수 있고, 상품의 후기를 통해서 상품 구매 결정에 참고할 수 있다. 회원은 자유게시판에 자유롭게 글을 쓸 수 있다. 회원은 자유게시판에서 글쓰기, 수정, 삭제를 할 수 있다.

**관리자 기능**

괸리자는 관리자 계정을 통해 사이트에 로그인하여 사용자들을 관리할 수 있다. 관리자는 모든 회원의 거래 또는 신고 정보를 볼 수 있으며 이 정보를 사용하여 신고당한 회원 또는 상품들을 처리할 수 있다. 관리자는 자유게시판에 공지를 쓸 수도 있고 회원이 작성한 부적절할 게시글을 삭제할 수도 있다.

**상품 등록/수정/삭제**

회원은 판매할 상품을 중고 장터에 등록할 수 있다. 등록된 상품은 회원이 수정 또는 삭제가 가능하다.

**상품 검색**

사용자가 구매를 원하는 상품을 검색할 수 있다. 구매할 상품의 이름, 제목을 검색할 수 있고 필터를 통하여 사용자가 원하는 옵션에 해당하는 상품들을 검색할 수 있다.

**판매 중인 상품/판매 내역 확인**

판매 중인 상품은 상품 조회 페이지에서 상품의 이미지, 제목, 가격, 상품 판매 지역, 거래 방법, 등록 시간 정보를 통해 확인할 수 있다. 해당 상품을 클릭하면 상품의 상세 정보를 확인할 수 있다. 회원의 상품 판매 내역은 마이페이지에서 자신이 어떤 상품을 판매했는지 확인할 수 있다.

**데이터 시각화**

시세 조회 페이지에서 특정 카테고리에 해당하는 상품들의 시세를 확인할 수 있다. 마이페이지에서 자신이 어떤 상품을 판매했고, 어떤 상품을 구매했는지 확인할 수 있다.

* 특별 기능

**아이디/비밀번호 찾기**

회원이 아이디/비밀번호를 잃어버렸을 때, 등록된 이메일로 인증번호를 전송하여, 본인확인을 통해 아이디/비밀번호를 찾을 수 있다.

**상품 판매 지도**

거래 지역을 설정하여 특정 지역에서 어떤 거래가 진행되는지 지역별로 확인할 수 있다.

**상품 추천**

좋아요를 많이 받은 상품들을 메인페이지에서 추천해준다.

**상품 상세 검색**

사용자가 회원가입을 할 때, 거주지역을 설정하게 하고, 상품을 등록할 때 희망거래장소 태그를 설정하게 한다. 이 정보를 이용하여 상품 상세 검색에서 지역별 판매 상품과 태그(카테고리)별로 상품을 상세하게 검색할 수 있다.

**사용자 평판(신뢰도)**

구매자가 상품의 후기를 작성할 때 별점을 줘서 판매자의 기존 평판(신뢰도)에 영향을 미칩니다.

**사용자/상품 신고**

적절하지 않은 상품이 판매 중일 때, 신고를 통해서 해당 상품/판매자를 신고할 수 있다. 이렇게 신고된 상품/판매자는 관리자가 보고 적절한 처리를 진행할 수 있다.

* 페이지 디자인

메인페이지

텍스트, 스크린샷, 도표, 평행이(가) 표시된 사진

자동 생성된 설명

상품조회 페이지

텍스트, 도표, 평행, 평면도이(가) 표시된 사진

자동 생성된 설명

판매중인 상품 상세정보 페이지

텍스트, 스크린샷, 도표, 평행이(가) 표시된 사진

자동 생성된 설명

판매완료 상품 상세정보 페이지

텍스트, 도표, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

상품등록 페이지

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

상품 수정/삭제 페이지

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

시세조회 페이지

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

마이페이지

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

회원정보수정

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

나의 판매 내역

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

나의 구매 내역

텍스트, 스크린샷, 폰트, 친필이(가) 표시된 사진

자동 생성된 설명

내가 찜한 상품

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

관리자 마이페이지

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

관리자 회원관리 페이지

텍스트, 스크린샷, 도표, 번호이(가) 표시된 사진

자동 생성된 설명

자유게시판

텍스트, 스크린샷, 번호, 도표이(가) 표시된 사진

자동 생성된 설명

게시판 글쓰기

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

게시판 글 수정/삭제

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

게시판 공지 쓰기

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

게시판 공지 수정/삭제

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

**4.ER diagram**

텍스트, 스크린샷, 도표, 평행이(가) 표시된 사진

자동 생성된 설명

usr의 필수 사용자(Id)와 item\_datas의 상품 등록자(Uid)는 비식별 관계로 매칭되며, 필수 사용자 ID는 한 명이지만, 상품 등록자는 한 명 이상이 될 수 있다.

usr의 필수 사용자(Id) 한 명은 user\_like\_items에서 찜한 사용자(Uid)와 식별 관계로 연결되며, 찜한 사용자는 최소 한 명 이상일 수 있다.

usr의 필수 사용자(Id) 한 명은 question에서 상품 문의자(C\_uid)와 비식별 관계로 연결되며, 문의자는 한 명 이상일 수 있다.

usr의 필수 사용자(Id) 한 명은 request에서 상품 구매 요청자(R\_uid)와 식별 관계로 연결되며, 구매 요청자는 한 명 이상일 수 있다.

usr의 필수 사용자(Id) 한 명은 review에서 상품 구매 후기를 쓴 사용자(B\_uid)와 식별 관계로 연결되며, 구매 후기를 남긴 사용자 수는 한 명 이상일 수 있다.

usr의 필수 사용자(Id) 한 명은 review에서 상품 판매자(S\_uid)와 비식별 관계로 연결되며, 상품 판매자는 한 명 이상일 수 있다.

usr의 필수 사용자(Id) 한 명은 report에서 신고당한 사용자(Rp\_uid)와 식별 관계로 연결되며, 신고당한 사용자는 한 명 이상일 수 있다.

usr의 필수 사용자(Id) 한 명은 board에서 작성자(Uid)와 식별 관계로 연결되며, 작성자는 최소 한 명 이상일 수 있다.

item\_datas에서 상품 판매자(Uid)가 한 명 이상일 경우, usr의 사용자(Id)는 필수적으로 비식별 관계로 매칭되며 한 명만 연결된다.

item\_datas의 필수적인 판매 상품(Ino) 한 개는 user\_like\_items에서 찜한 상품(Ino)과 식별 관계로 연결되며, 최소 한 개 이상일 수 있다.

item\_datas의 필수적인 판매 상품(Ino) 한 개는 question에서 문의된 상품(Ino)과 비식별 관계로 연결되며, 문의 상품은 한 개 이상일 수 있다.

item\_datas의 필수적인 판매 상품(Ino) 한 개는 request에서 구매 요청 상품(Ino)과 식별 관계로 연결되며, 요청된 상품은 최소 한 개 이상일 수 있다.

item\_datas의 필수적인 판매 상품(Ino) 한 개는 review에서 후기가 작성된 상품(Ino)과 식별 관계로 연결되며, 최소 한 개 이상일 수 있다.

item\_datas의 필수적인 판매 상품(Ino) 한 개는 report에서 신고당한 상품(Ino)과 식별 관계로 연결되며, 신고된 상품은 한 개 이상일 수 있다.

user\_like\_items에서 찜한 사용자(Uid)가 한 명 이상일 경우, usr의 사용자(Id)는 필수적으로 식별 관계로 한 명이 매칭된다.

user\_like\_items에서 찜한 상품(Ino)이 한 개 이상일 경우, item\_datas의 판매 상품(Ino)은 필수적으로 식별 관계로 한 개가 매칭된다.

question에서 문의한 사용자(C\_uid)가 한 명 이상일 경우, usr의 사용자(Id)는 필수적으로 비식별 관계로 한 명이 매칭된다.

question에서 문의한 상품(Ino)이 한 개 이상일 경우, item\_datas의 판매 상품(Ino)은 필수적으로 비식별 관계로 한 개가 매칭된다.

question의 필수적인 문의글(Qno) 한 개는 question에서 문의글에 대한 답변(Ans\_Qno)과 식별 관계로 연결되며, 문의글에 대한 답변은 한 개 이상일 수 있다.

request에서 구매를 요청한 사용자(R\_uid)가 한 명 이상일 경우, usr의 사용자(Id)는 필수적으로 식별 관계로 한 명이 매칭된다.

request에서 구매를 요청한 상품(Ino)이 한 개 이상일 경우, item\_datas의 상품(Ino)는 필수적으로 식별 관계로 한 개가 매칭된다.

review에서 상품의 후기를 단 사용자(S\_uid)가 한 명 이상일 경우, usr의 사용자(Id)는 필수적으로 식별 관계로 한 명이 매칭된다.

review에서 후기가 달린 상품의 판매자(B\_uid)가 한 명 이상일 경우, usr의 사용자(Id)는 필수적으로 비식별 관계로 한 명이 매칭된다.

Review에서 후기가 달린 상품(Ino)이 한 개 이상일 경우, item\_datas의 판매 상품(Ino)은 필수적으로 식별 관계로 한 개가 매칭된다.

report에서 신고당한 사용자(Rp\_uid)가 한 명 이상일 경우, usr의 사용자(Id)는 필수적으로 식별 관계로 한 개가 매칭된다.

report에서 신고당한 상품(Ino)가 한 개 이상일 경우, item\_datas의 판매 상품(Ino)는 필수적으로 식별 관계로 한 개가 매칭된다.

board에서 게시글 작성자(Uid)가 한 명 이상일 경우, usr의 사용자(Id)는 필수적으로 식별 관계로 한 명이 매칭된다.

**5.모든 테이블 설명**

1. db\_market

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

1. usr(Uname, Age, Sex, Phone, Email, Address, Id, Passwd, Auth, Nickname, Profile\_image, Rp\_cnt, Rating\_cnt, Avg\_rating)

* usr는 유저 정보를 담은 테이블입니다.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* Uname은 유저의 실명을 저장합니다.
* Age는 유저의 나이를 저장합니다.
* Sex는 유저의 성별을 저장합니다.
* Email은 유저의 이메일 주소를 저장합니다. 아이디/비밀번호 찾기에 활용됩니다.
* Address는 유저의 거주지 주소를 저장합니다. 위치기반 매물검색에 사용됩니다.
* Id는 유저의 아이디를 저장합니다.
* Passwd는 유저의 비밀번호를 저장합니다.
* Auth는 유저의 권한을 저장합니다. 권한으로 관리자와 일반유저를 구분 짓습니다. 관리자는 DB에 접근하여 데이터 삽입, 수정 및 삭제를 진행 할 수 있습니다. default 값은 0으로, 이는 일반유저임을 의미하고, 관리자가 유저의 Auth를 1로 설정하면, 해당 유저는 관리자 권한을 가지게 됩니다.
* Nickname은 유저의 닉네임을 저장합니다. 중고거래 사이트 내에서는 개인정보 보호를 위해 실명이 아닌 닉네임으로 활동하게 됩니다. UNIQUE이므로 유저 간 닉네임은 중복될 수 없고, Nullable이므로 작성하지 않는다면 익명으로 표시됩니다.
* Profile\_image는 유저의 프로필 사진을 저장합니다.
* Rp\_cnt는 유저가 신고받은 횟수를 저장합니다. 신고가 일정 횟수 이상 접수되면, 패널티가 적용됩니다.
* Rating\_cnt는 유저의 판매매물 리뷰글을 작성한 사람의 수를 저장합니다. 이는 별점평균을 집계하는데 사용됩니다.
* Avg\_rating은 유저의 별점평균을 저장합니다.
* 새로운 리뷰글이 작성되면 trigger가 작동하여

Avg\_rating = (Avg\_rating \* Rating\_cnt + new\_rating) / (Rating\_cnt + 1);

Rating\_cnt = Rating\_cnt + 1;

로 업데이트됩니다.

* 각 유저는 고유한 아이디를 가지므로, usr 테이블의 PK로 설정했습니다.

1. item\_datas(Uid, Ino, Title, Image, Price, Content, Deal\_way, Place, Like\_cnt, Reg\_date, B\_uid, Subimg1, Subimg2, Group1, Group2, Group3)

* item\_datas는 매물 판매글에 대한 정보를 담은 테이블입니다.

텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* Uid는 매물 판매글을 등록한 유저의 아이디를 저장합니다.
* Ino는 매물의 아이디를 저장합니다.
* Title은 매물 판매글의 제목을 저장합니다.
* Image는 매물의 이미지를 저장합니다. 해당 이미지는 판매글을 조회할 때 대표사진으로 보여질 것이므로 반드시 등록돼야합니다.
* Price는 매물의 가격을 저장합니다.
* Content는 매물 판매글의 본문을 저장합니다.
* Deal\_way는 매물의 거래방식을 저장합니다.
* Place는 거래 희망 장소입니다. NOT NULL이므로 입력하지 않는다면, 차후에 구매자와 판매자가 합의 하에 거래 장소를 정하게 됩니다.
* Like\_cnt는 유저들이 매물 판매글에 누른 좋아요 횟수입니다.
* Reg\_date는 매물 등록 날짜입니다.
* B\_uid는 매물 구매자 아이디입니다. default는 NULL로, 구매자가 해당 매물을 구매한다면 B\_uid에 구매자의 아이디가 저장됩니다. 따라서 B\_uid가 NULL일 때, 상품이 판매중임을 알 수 있습니다.
* Subimg1, Subimg2는 Image 외에 추가로 등록할 이미지입니다.
* Group1, Group2, Group3는 매물의 상품 분류입니다. Group1은 대분류, Group2는 중분류, Group3는 소분류입니다.
* Place, B\_uid, Subimg1, Subimg2를 제외하고는 모두 NOT NULL이므로 반드시 값이 입력돼야 합니다.
* 각 매물의 아이디는 AutoIncrement를 적용하여 고유한 값을 가지므로, item\_datas 테이블의 PK로 설정했습니다.

1. board(Uid, Title, Content, Image, Notice, Reg\_date)

* board는 게시판에 게시된 게시글에 대한 정보를 담은 테이블입니다.

텍스트, 폰트, 번호, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* Uid는 게시물을 쓴 유저의 아이디입니다.
* Title은 게시물의 제목입니다.
* Content는 게시물의 본문내용입니다.
* Image는 게시물에 게시된 이미지입니다.
* Notice는 게시물이 공지글인지를 나타냅니다. 공지글은 게시판 상단에 위치하기에 이를 판별하기 위해 사용됩니다.
* Reg\_date는 게시물을 작성한 날짜입니다.
* 같은 유저가 동시에 두 개의 게시글을 작성하는 것은 불가능하므로, (Uid, Reg\_date)를 PK로 설정했습니다.

1. question(Ino, Qno, Q\_uid, Content, Private, Reg\_date, Ans\_Qno)

* question은 매물에 대한 질문글과, 그에 대한 답변글에 대한 정보를 담은 테이블입니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* Ino는 매물의 아이디로, 어떤 매물에 대한 글인지를 나타내는 정보입니다.
* Qno는 글의 고유 아이디입니다.
* Q\_uid는 작성자의 아이디입니다.
* Content는 글 본문입니다.
* Private는 글의 공개범위입니다. 0은 공개, 1은 비공개입니다.
* Reg\_date는 글을 작성한 날짜입니다.
* Ans\_Qno는 답변글이 어떤 질문글에 대한 답변인지를 나타냅니다. 질문글과 답변글을 question 테이블에 같이 저장하기 때문에 질문글과 답변글을 구분하려는 목적도 있습니다. Ans\_Qno가 NULL이면 질문글이고, Ans\_Qno가 NULL이 아니면 답변글임을 알 수 있습니다.
* 각 글의 아이디는 AutoIncrement를 적용하여 고유한 값을 가지므로, question 테이블의 PK로 설정했습니다.

1. report(Ino, Rp\_uid, Content, Reg\_date)

* report는 매물에 대한 신고글 정보를 담고 있는 테이블입니다.

텍스트, 폰트, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

* Ino은 매물의 아이디입니다. 어떤 매물에 대한 신고글인지를 알 수 있습니다.
* Rp\_uid는 신고한 유저의 아이디입니다. 어떤 유저가 작성한 글인지를 알 수 있습니다.
* Content는 신고글 본문입니다. 신고사유 등을 저장합니다.
* Reg\_date는 신고글을 작성한 날짜입니다.
* 유저는 각 매물에 대해 한 번의 신고글 작성만이 가능합니다. 따라서 (Ino, Rp\_uid)를 PK로 설정했습니다.

1. review(Ino, S\_uid, B\_uid, Content, Ratings, Reg\_date)

* review는 매물에 대한 후기글 정보를 담고 있는 테이블입니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* Ino은 매물의 아이디입니다. 어떤 매물에 대한 후기글인지 알 수 있습니다.
* S\_uid는 판매자 아이디입니다.
* B\_uid는 구매자 판매자 아이디입니다.
* Content는 후기글 본문입니다.
* Ratings는 별점입니다. 1, 2, 3, 4, 5점 중에 고를 수 있습니다.
* Reg\_date는 후기글 작성 날짜입니다.
* 하나의 매물에는 구매자가 한 명만이 존재할 수 있으므로 (Ino, B\_uid)를 PK로 설정했습니다.

1. request(Ino, R\_uid, Content, Deal\_way, Location, Phone, Contact, Reg\_date)

* request는 매물에 대한 구매요청글 정보를 담고 있는 테이블입니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* Ino는 매물의 아이디입니다. 어떤 매물에 대한 구매요청인지 알 수 있습니다.
* R\_uid는 구매요청자의 아이디입니다. 어떤 유저가 구매요청을 한 것인지 알 수 있습니다.
* Content는 구매요청글 본문입니다.
* Deal\_way는 거래 방식입니다. 직거래와 택배거래 중 선택할 수 있습니다.
* Location은 거래 희망 장소입니다. Deal\_way가 직거래일 때 Location을 정할 수 있습니다.
* Phone은 전화번호입니다.
* Contact는 희망 연락 플랫폼입니다. ex)카카오톡, 이메일 등
* Reg\_date는 구매요청 날짜입니다.
* 각 유저는 하나의 매물에 대해 한 번의 요청만 보낼 수 있으므로, (Ino, R\_uid)를 PK로 설정했습니다.

1. user\_like\_item(Ino, Uid)

* user\_like\_item은 유저와 매물 간의 좋아요 관계 정보를 담은 테이블입니다. 어떤 유저가 어떤 매물에 좋아요를 눌렀는지 알 수 있습니다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* Ino은 매물의 아이디입니다.
* Uid는 유저의 아이디입니다.
* 각 유저는 한 매물에 대해 한 번의 좋아요만 누를 수 있으므로 (Ino, Uid)를 PK로 설정했습니다.

**6.사용되는 SQL문 설명**

|  |  |  |
| --- | --- | --- |
| No | SQL 문 종류 | SQL 문 |
| 1 | DELETE | DELETE FROM item\_datas  WHERE Uid = ‘user8’ AND Title = ‘item8-3’; |
| item\_datas 테이블에서 Uid가 ‘user8’이고 Title이 ‘item8-3’인 것을 삭제 |
| 상품 판매 테이블에서 등록된 상품을 삭제할 때 사용한다. |
| 2 | UPDATE | UPDATE usr  SET Passwd = ‘NEWpassword2’  WHERE Id = ‘user2’; |
| usr 테이블에서 Id가 ‘user2’인 튜플을 찾아서 해당 튜플의 Passwd를 ‘NEWpassword2’로 업데이트 |
| 사용자의 비밀번호를 변경할 때 사용한다. |
| 3 | SELECT | SELECT Ino, Title, Image, Price, Like\_cnt  FROM item\_datas  WHERE Title LIKE ‘%Item1-%’; |
| item\_datas 테이블에서 Title이 “Item1-”을 포함하는 문자열을 가지는 튜플들의 Ino, Title, Image, Price, Like\_cnt값을 선택 |
| 판매 상품들 중에서 제목이 “Item1-”로 시작하는 상품을 검색할 때 사용한다. |
| 4 | BETWEEN | SELECT Title, Price, Deal\_way, Image  FROM item\_datas  WHERE Price BETWEEN 10000 AND 25000; |
| item\_datas 테이블에서 Price 값이 10000과 25000 사이의 값을 가지는 튜플들의 Title, Price, Deal\_way, Image 값을 선택 |
| 판매 상품들 중에서 가격이 10000에서 25000 사이인 상품들을 검색할 때 사용한다. |
| 5 | NULL | SELECT Title, Price, Deal\_way, Image  FROM item\_datas  WHERE B\_uid IS NOT NULL; |
| item\_datas 테이블에서 B\_uid가 NULL이 아닌 튜플들의 Title, Price, Deal\_way, Image 값들을 선택 |
| B\_uid의 값이 NULL이 아닌 것은 거래가 완료된 상품을 말한다. 따라서 거래가 완료된 상품들을 검색할 때 사용한다. |
| 6 | ORDER BY | SELECT i.Image, i.Title, i.Content, i.Reg\_date  FROM usr AS U JOIN item\_datas AS I ON u.Id = i.B\_uid  WHERE u.Id = ‘user1’  ORDER BY i.Reg\_date DESC; |
| usr 테이블과 item\_datas 테이블을 usr의 Id 값과 item\_datas의 Id 값이 같은 것을 기준으로 INNER JOIN을 해서 usr의 Id 값이 ‘user1’인 튜플들을 선택해서 그 튜플들을 Reg\_date값을 기준으로 내림차순 정렬한 후, 그 튜플들의 Image, Title, Content, Reg\_date 값들을 선택 |
| 사용자가 자신이 구매한 상품들을 최신순으로 검색할 때 사용한다. |
| 7 | EXIST와 DISTINCT | SELECT DISTINCT u.Id, u.Uname  FROM usr AS u  WHERE EXISTS (  SELECT \*  FROM report AS r  JOIN item\_datas as i ON r.Ino = i.Ino  WHERE i.Uid = u.Id  ); |
| report 테이블과 item\_datas 테이블을 report의 Ino 값과 item\_datas의 Ino 값이 같은 튜플들로 INNER JOIN하고, 그 튜플들 중에서 item\_datas의 Uid와 report의 Id가 같은 투플이 존재할 때, 그 튜플들의 Id, Uname 값들을 중복없이 선택 |
| 사용자들 중에서 한 번이라도 신고당한 사용자를 검색할 때 사용한다. |
| 8 | VIEW | CREATE VIEW forsale AS  SELECT Uid, Ino, Title, Image, Price, Deal\_way, Place, Like\_cnt, Reg\_date  FROM item\_datas;  SELECT \*  FROM forsale  ORDER BY Reg\_date DESC, Price ASC; |
| item\_datas 테이블에서 Uid, Ino, Title, Image, Price, Deal\_way, Place, Like\_cnt, Reg\_date 값들을 선택해서 새로운 ‘forsale’ view를 만듦  ‘forsale’ view를 Reg\_date 값을 기준으로 내림차순 정렬하고, Price 값을 기준으로 오름차순 정렬하고 모든 튜플들을 선택한다. |
| 판매 상품들 중에서 판매 상품 목록 조회에 필요한 column 들만 추출해서 ‘forsale’ view를 만들 때 사용한다.  만들어진 view는 등록된 시간이 최신순부터(등록된 시간이 동일하다면 가격이 적은 것부터) 판매 상품을 보여줄 때 사용할 수 있다. |
| 9 | IN | SELECT i.Ino, i.Image, i.Title, i.Price, i.Deal\_way, i.Reg\_date  FROM item\_datas AS i  WHERE i.Ino IN(SELECT k.Ino  FROM user\_like\_items AS k  WHERE i.Ino = k.Ino AND i.Place = ‘Seoul’)  ORDER BY i.Like\_cnt DESC, i.Price ASC; |
| user\_like\_items테이블에서 item\_datas의 Ino값과 user\_like\_items의 Ino 값이 같고 item\_datas의 Place값이 ‘Seoul’인 튜플들의 Ino값들을 선택한다. 이 튜플들 중에서 item\_datas의 Ino에 해당하는 튜플들의 Ino, Image, Title, Price, Deal\_way, Reg\_date 값들을 선택한다. 그 튜플들을 Like\_cnt 값을 기준으로 내림차순 정렬하고, Like\_cnt값이 같다면 Price 값을 기준으로 오름차순 정렬한다. |
| 사용자들이 찜한 판매 상품들 중에서 판매 지역이 ‘Seoul’인 상품들을 좋아요가 많은 순으로 정렬하고 같은 수라면 가격이 적은 순으로 정렬해서 검색할 때 사용한다. |
| 10 | SOME | SELECT u.Id, u.Uname  FROM usr AS u  WHERE u.Id = SOME (  SELECT B\_uid  FROM item\_datas  WHERE Group1 = ‘Books’  AND B\_uid IS NOT NULL  ); |
| item\_datas 테이블 중에서 Group1의 값이 ‘Books’이고, B\_uid의 값이 NULL이 아닌 튜플들이 하나라도 존재하면 그 튜플들 중에서 usr 테이블의 Id 값과 같은 튜플들의 Id값과 Uname값들을 선택 |
| 판매된 ’Books’ 상품들 중에서 한 번이라도 ‘Books’을 산 사용자들의 목록을 검색할 때 사용한다. |
| 11 | ALL | SELECT Ino, Title, Like\_cnt, Place  FROM item\_datas  WHERE Place = ‘Seoul’  AND Like\_cnt >= ALL (  SELECT Like\_cnt  FROM item\_datas  WHERE Place = ‘Seoul’  ); |
| item\_datas 테이블에서 Place의 값이 ‘Seoul’이고 item\_datas 테이블에서 Place의 값이 ‘Seoul’인 튜플들 중에서 LIke\_cnt값을 선택하고, 이 모든 값들보다 크거나 같은 Like\_cnt를 가진 튜플의 Ino, Title, Like\_cnt, Place 값을 선택 |
| Place 값이 ‘Seoul’인 판매 상품들 중에 가장 많은 좋아요를 받은 판매 상품을 검색하는데 사용된다. |
| 12 | RIGHT OUTER JOIN | SELECT q.Ino, q.Q\_uid, q.Content AS Question, a.Q\_uid AS  Answer\_uid, a.Ans\_Qno, a.Content AS Answer  FROM question AS a  RIGHT OUTER JOIN question AS q ON a.Ans\_Qno = q.Qno  WHERE q.Ans\_Qno IS NULL; |
| question 테이블과 또 다른 question테이블을 한 question 테이블의 Ans\_Qno값과 다른 question 테이블의 Qno값이 같은 튜플들을 기준으로 RIGHT OUTER JOIN을 한다. 이렇게 만들어진 Relation중에서 Ans\_Qno의 값이 NULL인 튜플들의 Ino, Q\_uid, Content 값들을 선택해 ‘Question’으로 반환하고, Q\_uid를 ‘Answer\_uid’로 반환하고, Ans\_Qno값을 반환하고 Content값을 ‘Answer’로 반환한다. |
| 문의글에서 답변이 달리지 않은 문의글만 검색할 때 사용한다. |
| 13 | UNION | ( SELECT Title, Price, Group1, Group2  FROM item\_datas  WHERE Group1 = ‘Electronics’ AND Group2 = ‘Phones’)  UNION  ( SELECT Title, Price, Group1, Group2  FROM item\_datas  WHERE Group1 = ‘Electronics’ AND Group2 = ‘Tablets’) |
| item\_datas 테이블에서 Group1값이 ‘Electronics’이고 Group2값이 ‘Phones’인 튜플들의 Title, Price, Group1, Gropu2값들을 선택하고, item\_datas 테이블에서 Group1값이 ‘Electronics’이고 Group2값이 ‘Tablets’인 튜플들의 Title, Price, Group1, Group2값들을 선택한다. 선택된 두 튜플들로 UNION을 하여 새로운 튜플들을 만든다. |
| 판매 상품들 중에서 Group1의 값이 ‘Electronics’인 상품들 중에서 Group2의 값이 ‘Phone’이거나 ‘Tablets’인 상품들을 함께볼 때 사용한다. |
| 14 | INSERT | INSERT INTO usr(Uname, Age, Sex, Phone, Email, Address, Id, Passwd, Auth, Nickname, Profile\_image, Rp\_cnt, Rating\_cnt, Avg\_rating)  VALUES  (…); |
| 본 프로젝트에서 사용자 정보에 새로 가입한 사용자를 추가할 때 사용 |
| usr 테이블에 이름, 나이, 성, 전화번호, 이메일, 주소, 아이디, 비밀번호, 권한, 닉네임, 프로필사진, 리뷰 개수, 별점, 별점평균 값을 가진 새로운 row를 추가 |
| 15 | TRIGGER | CREATE TRIGGER ratingUpgrade  ALTER INSERT ON review  FOR EACH ROW  BEGIN  …  END; |
| review 테이블에 새로운 row가 추가된 후, usr 테이블의 Rating\_cnt, Avg\_rating을 재계산 |
| 본 프로젝트에서 새로운 리뷰가 추가될 때, usr 테이블의 Rating\_cnt를 1 증가시키고, Avg\_rating을 재계산 합니다. |
| 16 | GROUP BY | SELECT Group1, AVG(Price)  FROM item\_datas  GROUP BY Group1 |
| item\_datas 테이블에서 Group1 column값이 같은 row끼리 그룹지음. |
| 본 프로젝트에서 Group 별로 매물의 평균가격을 구할 때 사용 |
| 17 | AGGREGATION HAVING | SELECT Address, COUNT(\*)  FROM usr  WHERE Avg\_rating >= 3 AND  Address IN ( SELECT Address  FROM usr  GROUP BY Address  HAVING COUNT(\*) >= 2) GROUP BY Address; |
| usr 테이블에서 Address별로 거주자가 2명 이상인 주소만 필터링 |
| 2명 이상이 거주하는 지역에 한해, 지역별로 평균 별점이 3점 이상인 사람들을 지역별로 거주지 수와 함께 select |
| 18 | CASE | UPDATE usr  SET Avg\_rating =  CASE  WHEN Address LIKE ‘Daejeon%’ THEN 2  WHEN Address LIKE ‘Ulsan%’ THEN 4  WHEN Address LIKE ‘Busan%’ THEN 3  WHEN Address LIKE ‘Incheon%’ THEN 5  WHEN Address LIKE ‘Daegu%’ THEN 1  WHEN Address LIKE ‘Gwangju%’ THEN 2  ELSE 4  END; |
| Address 별로 값을 다르게 설정 |
| 각 도시 별로 별점평균을 다르게 설정 |
| 19 | JOIN USING | SELECT Title, Image, Price, R.Content, Ratings  FROM item\_datas I JOIN review R using(Ino)  WHERE Ratings >= 4; |
| item\_datas 테이블과 review 테이블을 JOIN한 결과에서 Ino가 같은 row만 선택 |
| 리뷰별점이 4점 이상인 매물에 대하여 판매글의 제목, 이미지, 가격 그리고 리뷰글의 본문내용, 별점을 선택 |
| 20 | LEFT JOIN | SELECT Uname, Age, Sex  FROM usr  WHERE id IN  (SELECT Q\_uid  FROM question Q LEFT JOIN request R ON (Q.Q\_uid = R.R\_uid AND Q.Ino = R.Ino)  WHERE R.R\_uid IS NULL AND R.Ino IS NULL AND Q.Ans\_Qno IS NULL); |
| 하나의 매물에 대해 동일한 유저가 질문과 구매요청을 한 매물에 대해 question 테이블과 request 테이블을 LEFT JOIN 합니다. |
| JOIN은 동일한 유저가 질문과 구매요청을 한 매물에 대해 조건이 걸려있지만, LEFT JOIN이므로 동일한 유저가 질문은 했지만 구매요청은 하지 않은 매물 또한 JOIN의 결과로 남게됩니다. 따라서 위의 쿼리는 질문은 했지만 구매요청은 하지 않은 유저의 실명, 이름, 성을 선택합니다. |
| 21 | NOT EXISTS | SELECT U.Uname, U.Age, U.Sex  FROM usr U  WHERE NOT EXISTS  (SELECT \*  FROM request R  WHERE R\_uid = U.Id); |
| 특정 유저가 쓴 리뷰글이 존재하는지 여부를 판단 |
| 리뷰글을 쓴 적 없는 유저의 실명과 나이, 성을 선택합니다. |
| 22 | SELF JOIN | SELECT Q.Content AS Question, A.Content AS ANSWER  FROM question AS Q, question AS A  WHERE Q.Ino = 15 AND A.Ans\_Qno = Q.Qno; |
| question 테이블은 질문글과 답변글을 함께 저장하므로 질문글에 관한 question 테이블과 답변글에 관한 question 테이블을 JOIN |
| 15번 매물에 대한 질문글과 해당 질문에 대한 답변글이 존재한다면, 해당 질문글과 답변글의 본문내용을 선택 |
| 23 | CONCAT | SELECT Nickname, CONCAT('신고 당한 횟수: ', CONCAT(Rp\_cnt, CONCAT(', 지역: ', CONCAT(Location, CONCAT(', 내용: ', Content))))) AS request\_info  FROM request JOIN usr ON(R\_uid = Id); |
| 각 값들에 대해 어떤 column에 대한 내용인지 CONCAT을 통해 나타냄 |
| 구매요청을 한 유저들의 정보(닉네임, 신고 당한 횟수, 지역)와 구매요청 본문내용을 선택 |
| 24 | WITH | WITH LARGECATEGORY(Group1) AS  (SELECT Group1  FROM item\_datas  GROUP BY Group1  HAVING COUNT(\*) > 3)  SELECT Group1, COUNT(\*)  FROM item\_datas  WHERE Price < 50000 AND Group1 IN  (SELECT Group1 FROM LARGECATEGORY)  GROUP BY Group1; |
| 대분류 별로, 매물의 개수가 4개 이상인 대분류 명을 임시테이블로 저장합니다. |
| 매물이 4개 이상 있는 대분류를, 가격이 50000 미만인 매물 개수와 함께 선택 |

**7.각 SQL문 실행 화면 캡처**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | SQL 문 종류 | | 실행 화면 | |
| 1 | DELETE | | 텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진  자동 생성된 설명 | |
| 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 | |
| DELETE 쿼리를 사용하여 item\_datas테이블에서 회원이 등록한 상품을 삭제하는 예제이다. | |
| 2 | UPDATE | | 텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진  자동 생성된 설명 | |
| 스크린샷, 텍스트, 폰트, 라인이(가) 표시된 사진  자동 생성된 설명 | |
| UPDATE 쿼리를 사용하여 아이디가 ‘user2’ 회원의 비밀번호를 업데이트하는 예제이다. | |
| 3 | SELECT | | 텍스트, 폰트, 번호, 라인이(가) 표시된 사진  자동 생성된 설명 | |
| 텍스트, 폰트, 번호, 라인이(가) 표시된 사진  자동 생성된 설명 | |
| SELECT 쿼리로 item\_datas 테이블에서 판매 상품의 제목에 ‘Item1-’가 들어가는 판매 상품의 정보를 가져오는 에제이다. | |
| 4 | | BETWEEN | | 텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진  자동 생성된 설명 |
| BETWEEN 쿼리를 사용하여 item\_datas 테이블에서 가격이 10000에서 25000 사이인 판매 상품을 검색하는 예제이다. |
| 5 | | NULL | | 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 |
| NULL이 들어간 쿼리를 사용하여 item\_datas 테이블에서 전체 판매 상품들 중에서 거래가 완료된 상품을 검색하는 예제이다.  item\_datas에서 B\_uid는 NULL값일 때는 구매한 회원이 정보가 없으므로 판매 중인 상품이고, NULL값이 아니면 구매한 회원이 존재하므로 판매된 상품이다. |
| 6 | | ORDER BY | | 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 |
| ORDER BY를 사용한 쿼리를 사용하여 아이디가 ‘user1’인 회원의 상품 구매 목록을 최신순으로 검색하는 예제이다. |
| 7 | | EXIST와 DISTINCT | | 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 |
| EXIST와 DISTINCT를 이용한 쿼리를 사용하여 회원 중에서 한 번이라도 신고당한 회원의 Id와 Uname을 검색하는 예제이다. |
| 8 | | VIEW | | 텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진  자동 생성된 설명 |
| VIEW를 사용한 쿼리로 item\_datas 테이블에서 판매 상품 목록 조회에 필요한 column들만 검색하여 ‘forsale’ view를 만드는 예제이다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9 | IN | | 텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진  자동 생성된 설명 | |
| 텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진  자동 생성된 설명 | |
| IN을 이용한 쿼리를 사용하여 판매 상품들 중에서 거래 지역이 ‘Seoul’인 상품들을 좋아요가 많은 순으로 검색하는 예제이다. | |
| 10 | SOME | | 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 | |
| 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 | |
| SOME을 이용한 쿼리를 사용하여 구매 상품 중에서 ‘Books’을 한 번이라도 구매한 회원의 Id와 Uname을 검색하는 예제이다. | |
| 11 | ALL | | 텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진  자동 생성된 설명 | |
| 텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진  자동 생성된 설명 | |
| ALL을 이용한 쿼리를 사용하여 판매 상품들 중에서 거래 지역이 ‘Seoul’인 상품들 중에서 좋아요가 제일 많은 상품을 검색하는 예제이다. | |
| 12 | RIGHT OUTER JOIN | | 텍스트, 폰트, 번호, 소프트웨어이(가) 표시된 사진  자동 생성된 설명 | |
| 텍스트, 폰트, 번호, 소프트웨어이(가) 표시된 사진  자동 생성된 설명 | |
| RIGHT OUTER JOIN을 사용한 쿼리로 question 테이블에서 문의글만 검색하는 예제이다.  question 테이블은 문의글이 문의글 ‘Qno’와 문의글의 답글 ‘Ans\_Qno’이 있다. ‘Ans\_Qno’도 문의글이므로 question 테이블에 저장되지만 답글이 안 달린 경우에 NULL값을 가질 수 있다. 여기서 모든 문의글을 검색할 수 있지만 문의글의 답글이 마찬가지로 question 테이블에 저장되므로 같은 문의글이 생기게 된다. ㄸ라서 중복된 문의글을 없애기 위해서 WHERE 절에서 q.Ans\_Qno가 NULL인 튜플중에서 값을 검색한다. | |
| 13 | UNION | | 텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진  자동 생성된 설명 | |
| 텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진  자동 생성된 설명 | |
| UNION을 사용한 쿼리로 item\_datas 테이블에서 판매 상품의 카테고리가 ‘Electronics’이고 ‘Phones’인 판매 상품과 카테고리가 ‘Electronics’이고 ‘Tablets’인 판매 상품을 UNION해서 검색하는 예제이다. | |
| 14 | | INSERT | | 텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 |
| INSERT 쿼리를 사용해 usr 테이블에 회원을 추가하는 예제입니다. |
| 15 | | TRIGGER | | 텍스트, 스크린샷, 폰트, 디스플레이이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 폰트이(가) 표시된 사진  자동 생성된 설명  텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 |
| Trigger를 통해 리뷰글이 추가될 때마다 유저의 별점 평균을 업데이트 하는 예제입니다.  user1이 판매한 물건들에 대한 리뷰가 3개 추가되자, user1의 Rating\_cnt이 3이 되고, 별점평균은 (4+2+5)/3 = 3.66667이 됩니다. |
| 16 | | GROUP BY | | 텍스트, 폰트, 화이트, 디자인이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 스크린샷, 번호이(가) 표시된 사진  자동 생성된 설명  텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 |
| GROUP BY를 통해 대분류로 그룹화하고, 그룹별 평균 가격을 구하는 예제입니다.  대분류를, 대분류 별 매물 평균가격과 함께 선택합니다. |
| 17 | | AGGREGATION HAVING | | 텍스트, 폰트, 스크린샷이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 폰트, 라인, 번호이(가) 표시된 사진  자동 생성된 설명 |
| GROUP BY로 거주지 별로 유저를 나누고, HAVING으로 해당 거주지에 거주하는 유저가 2명 이상인 주소만 선택합니다.  따라서 거주자가 2명 이상인 주소와, 해당 주소에서 별점평균이 3점 이상인 유저의 수를 선택하게 됩니다. |
| 18 | | CASE | | 텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 번호, 폰트, 스크린샷이(가) 표시된 사진  자동 생성된 설명 |
| 데이터 초기 상태 초기화를 위해 CASE문을 통해 각 지역별로 유저의 별점평균을 다르게 설정했습니다.  결과이미지를 보면 각 도시별로 대전은 2점, 울산은 4점, 부산은 3점, 인천은 5점, 대구는 1점, 광주는 2점, 이외에 서울은 4점인 것을 확인할 수 있습니다. |
| 19 | | JOIN USING | | 텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 폰트, 번호, 스크린샷이(가) 표시된 사진  자동 생성된 설명텍스트, 폰트, 스크린샷이(가) 표시된 사진  자동 생성된 설명 |
| USING을 통해 같은 Ino를 가지는 매물과 리뷰를 JOIN합니다. 이후 별점이 4점 이상인 판매글의 제목,이미지,가격, 리뷰글의 본문, 별점을 선택합니다.  결과이미지를 보면 별점이 4점 이상인 22, 24, 31번 매물에 대해 선택된 것을 볼 수 있습니다. |
| 20 | | LEFT JOIN | | 텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진  자동 생성된 설명 |
|  |
| 질문은 했지만 구매요청은 하지 않은 유저의 실명, 이름, 성을 선택합니다.  같은 Ino에 대해서 question테이블과 request테이블에 둘 다 이름이 올라간 유저는 user1, 8입니다.  결과 이미지를 보면 question테이블에 존재하는 user1, 2, 3, 4, 7, 8 중에서 1, 8을 제외한 user 2, 3, 4, 7만이 선택되는 것을 확인할 수 있습니다. |
| 21 | | NOT EXISTS | | 텍스트, 폰트, 스크린샷이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 텍스트, 폰트, 번호, 라인이(가) 표시된 사진  자동 생성된 설명 |
| 특정 유저가 쓴 구매요청을 했는지를 NOT EXISTS를 통해 판별합니다.  구매요청 리스트에 존재하는 유저1, 3, 9, 10은 결과이미지에 존재하지 않는 것을 볼 수 있습니다. |
| 22 | | SELF JOIN | | 텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 폰트, 라인, 번호이(가) 표시된 사진  자동 생성된 설명 |
| 15번 매물에 대한 질문과 답변을 SELF JOIN을 통해 선택한 것을 확인할 수 있습니다. |
| 23 | | CONCAT | | 텍스트, 폰트, 스크린샷이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진  자동 생성된 설명 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명  텍스트, 폰트, 번호, 라인이(가) 표시된 사진  자동 생성된 설명 |
| 구매요청을 한 유저들의 닉네임과 정보, 구매요청글 본문을 한 문장으로 표현하기 위해 CONCAT을 사용했습니다.  구매요청 테이블에 존재하는 유저들에 대해 정보가 한문장으로 표현되는 것을 확인할 수 있습니다. |
| 24 | | WITH | | 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트, 스크린샷, 번호이(가) 표시된 사진  자동 생성된 설명 텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진  자동 생성된 설명 |
| 대분류 별로, 매물의 개수가 4개 이상인 대분류 명을 WITH를 통해 임시테이블로 저장합니다. 매물이 4개 이상 있는 대분류를, 가격이 50000 미만인 매물 개수와 함께 선택  매물의 개수가 4개 이상인 대분류는 전자, 북, 옷이고, 이 대분류들 중 50000원 미만인 매물은 각각 4, 5, 5개인 것을 확인할 수 있습니다. |