09조 레포트

2271006 강민서
2276103 도연수
2276184 안채연
2276188 양인경
2271063 하지연

(1) 선택한 기관의 데이터베이스 필요성 설명 및 요구 분석

필요성 설명

1. 항공기 정보

다양한 항공기 정보는 고객에게 보다 많은 선택지를 제공하여 최적의 항공편을 찾을 수 있게 합니다.

요구 데이터: 항공사이름, 항공기ID, 수용인원

2. 항공편 스케줄 및 가격 정보

고객이 여행 날짜와 목적지를 입력하면, 해당 조건에 맞는 항공편 정보를 제공해야 하므로 정확한 스케줄과 가격 정보가 필수적입니다.

요구 데이터: 소요시간, 출발지 및 도착지, 마일리지, 예약번호, 항공편 번호, 항공 요금

3. 마이페이지

간편한 회원가입 절차를 통해 고객번호로 식별된 고객에게 맞춤형 서비스를 제공하기 위해 필요합니다. 또한, 예약 및 결제 과정의 효율적인 처리를 위해서도 중요합니다.

요구 데이터: 고객 개인정보 (이름, 연락처, 생년월일, 아이디), 고객번호 및 로그인 정보, 과거 예약 내역, 결제 정보

4. 예약 관리 및 마일리지 관리

고객이 마일리지를 효율적으로 관리하고 사용할 수 있도록, 마일리지 적립 및 사용 정보를 제공해야 합니다.

요구 데이터: 마일리지 적립 내역, 마일리지 사용 가능 여부, 마일리지 잔액

5. 쿠폰 할인 혜택

고객이 5번 예약 확정마다 랜덤 할인 쿠폰을 발급받기 때문에, 이를 관리하고 적용하는 시스템이 필요합니다.

요구 데이터: 발급된 쿠폰 내역, 쿠폰 사용 가능 여부, 쿠폰 적용 시 할인 금액 및 조건

6. 검색 및 필터링 기능

고객이 항공편을 출발시간순 또는 가격순으로 정렬하여 선택할 수 있게 하기 위해, 검색 및 필터링 시스템이 필요합니다.

요구 데이터: 고객 검색 조건 (여행 날짜, 목적지, 출발지), 필터링 및 정렬 옵션 (출발 시간, 가격)

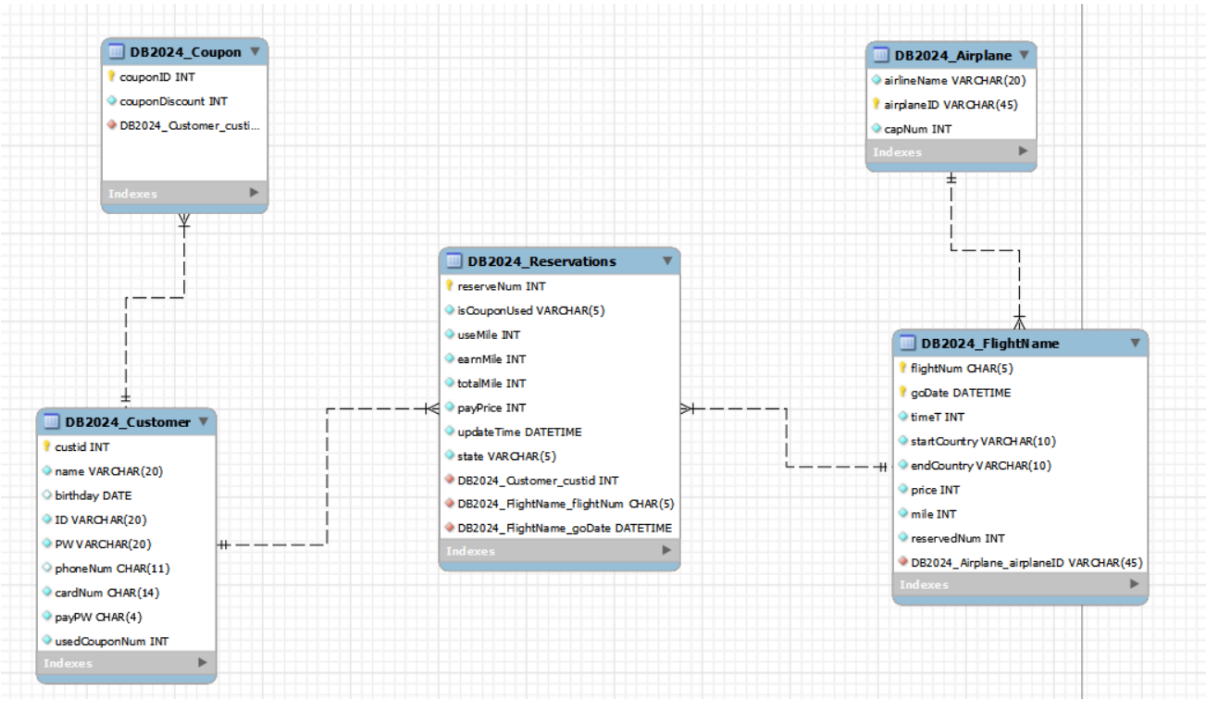
요구 분석

1) 데이터 정확성 및 최신성: 항공편 스케줄과 가격 정보는 실시간으로 업데이트되어야 합니다. 이를 위해 항공사와의 연동이 필요합니다.

2) 보안 및 개인정보 보호: 고객의 개인정보와 결제 정보는 철저하게 보호되어야 하며, 이에 따른 보안 시스템이 필요합니다.

3) 통합성: 마일리지 및 쿠폰 시스템이 각 항공사와 통합되어 일관되게 운영될 수 있어야 합니다.

(2) ER 다이어그램과 간단한 설명



개체는 DB2024_Customer, DB2024_Coupon, DB2024_FlightName, DB2024_Airplane로 총 4개이다. 4개의 개체 모두 강한개체이다.

1) DB2024_Customer와 DB2024_Coupon

- DB2024_Customer는 DB2024_Coupon을 보유하고 있고, DB2024_Coupon은 DB2024_Customer에 보유되고 있는 비식별자 일대다의 '보유'관계이다.

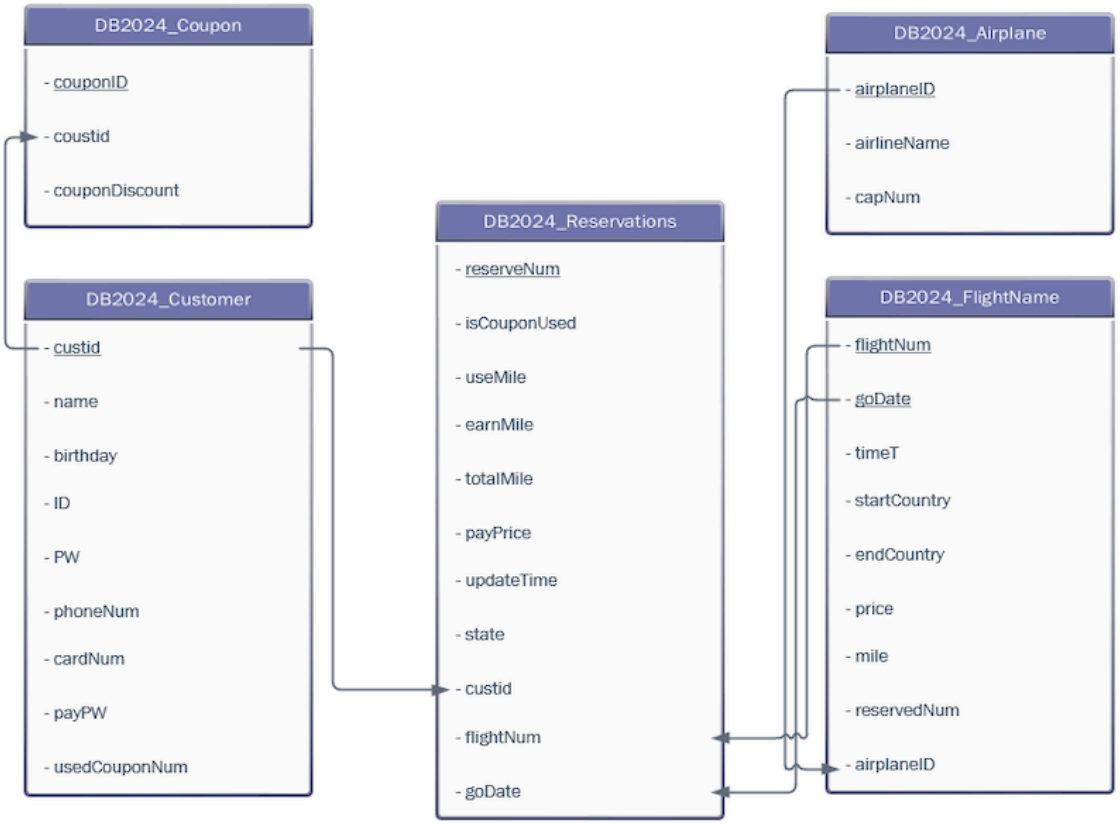
2) DB2024_Customer와 DB2024_FlightName

- DB2024_Customer과 DB2024_FlightName의 관계 타입은 DB2024_Reservations이다.
- 이진 관계이며, 일대다 관계이다.

3) DB2024_Airplane와 DB2024_FlightName

- DB2024_Airplane은 DB2024_FlightName을 포함하고, DB2024_FlightName은 DB2024_Airplane에 소속되는 비식별자 일대다의 '소속'관계이다.

(3) 데이터베이스 스키마 다이어그램. 테이블과 컬럼 그리고 관계(relationoships)들을 보여준다 MS Visio, MySQL Workbench 등을 사용해서 작성할 수 있다.



(4) 자바 코드의 클래스(class)와 메서드(method)에 대한 설명

클래스는 Main 클래스 하나를 사용하였다.

화면을 출력하는 함수와 기능을 위한 함수 등을 작성하여 사용하였다.

먼저 화면을 구성하기 위한 함수를 작성하였다.

head 함수는 현재 화면이 어떤 화면인지 가독성 높게 출력하는 함수이다.

login 함수는 로그인 화면을 보여주는 함수이다. 아이디와 비밀번호를 받아 로그인 기능을 수행할 수 있다. 로그인에 실패하면 다시 login 함수를 호출하고 로그인에 성공하면 home 함수를 호출한다.

home 함수는 홈 화면을 보여주는 함수이다. 마이페이지와 예매하기 화면으로 이동할 수 있는 버튼을 누를 수 있다.

myPage 함수는 마이페이지 화면을 보여주는 함수이다. 내 정보, 예약 내역, 마일리지, 쿠폰, 돌아가기 버튼을 누를 수 있다.

myInfo 함수는 내 정보 화면을 보여주는 함수이다. DB2024_Customer 릴레이션을 이용하여 고객 정보를 출력한다. 그 이후 정보 수정, 마이페이지 화면으로 갈 수 있는 버튼을 누를 수 있다.

changeMyInfo 함수는 내 정보 수정 화면을 보여주는 함수이다. 새로운 비밀번호, 전화번호, 카드번호, 결제비밀번호를 받아 정보를 수정한다. 수정 후에는 myInfo로 돌아간다.

reserved 함수는 내 예약 화면을 보여주는 함수이다. printReserved 함수를 사용하여 예약 내역을 출력한다. 그 이후 예약취소, 마이페이지 화면으로 이동할 수 있는 버튼을 누를 수 있다.

cancelReserve 함수는 예약 취소하기 화면을 보여주는 함수이다. printReserved를 이용하여 예약 내역을 출력한다. 그 이후 예약 취소를 진행할 예약번호를 입력받고 해당 번호가 예약 내역에 존재한다면 예약 취소를 진행한다. 취소 과정은 먼저 마일리지와

금액을 환불하고, 해당 예약이 쿠폰 발급 조건을 충족시키는 예약이었다면 쿠폰 회수를 진행한다.

마일리지와 금액 환불 과정은 다음과 같다. 먼저 입력 받은 번호가 예약 내역에 존재하는지 확인한다. 존재하는 경우 쿠폰을 사용했는지 확인한다. 만약에 쿠폰을 사용했으면 취소가 불가능함을 출력하고 다시 예약하기 화면으로 돌아간다. 쿠폰을 사용하지 않았다면 그 예약에서 사용한 마일리지, 적립한 마일리지, 현재 누적 마일리지를 받아와 계산을 통해 마일리지를 복구한다. 또, 그때 사용한 가격을 받아와 환불한 가격을 출력한다.

쿠폰 회수 과정은 다음과 같다. retrieveCoupon 함수를 통해 해당 예약이 쿠폰 발급 조건을 충족시키는 예약인지 확인한다. 만약 맞다면 예약을 취소했으니 가장 오래된 쿠폰을 삭제한다.

모든 기능을 수행 후 reserved로 돌아간다.

mileage 함수는 마일리지 사용 내역 화면을 보여주는 함수이다. DB2024_Reservations 릴레이션과 DB2024_FlightName 릴레이션을 이용하여 편명, 출발일, 사용 마일리지, 적립 마일리지를 출력한다. 이후 myPage로 돌아간다.

coupon 함수는 내 쿠폰 화면을 보여주는 함수이다. printCoupon 함수를 사용하여 사용가능한 쿠폰목록을 출력하고 home으로 돌아간다.

reserve 함수는 예약하기 화면을 보여주는 함수이다. 출발지, 도착지, 출발날짜, 도착날짜 등 여행 정보들을 받아 resultFilight 함수의 인자로 넣어 호출한다.

resultFlight 함수는 편명 검색 결과 및 예약 화면을 보여주는 함수이다. 시간순, 가격순, 예매, 돌아가기 버튼을 누를 수 있다. 시간순, 가격순을 누르면 이에 맞는 정렬에 맞추어 printFlight를 사용해 검색 결과를 출력하고 다시 resultFilght를 호출한다. 예매를 누르면 예매할 편명 선택하기 화면으로 이동한다.

select 함수는 예매할 편명 선택하기 화면을 보여주는 함수이다. 예약할 편명을 입력 받는다. 먼저 받은 편명이 목록에 존재하는지 확인한 후 이미 예약한 편명인지도 확인한다. 편명 릴레이션에도 존재하고 예약한 적이 없는 릴레이션이면 예매 페이지로 이동한다.

reserving 함수는 편명 예매하기 화면을 보여주는 함수이다. 먼저 고객 정보를 출력한다.

그리고 진행할 예약의 정보를 출력한다. 다음은 쿠폰 사용 유무를 묻고 사용한다면 사용가능한 쿠폰을 printCoupon 함수를 통해 출력하고 사용할 쿠폰 ID를 입력 받아 사용한다. 또, 마일리지 사용 여부를 묻고 선택에 사용한다면 얼마를 사용할 것인지 입력받아 그 값만큼 결제할 값을 할인한다. 마지막으로 입력한 쿠폰과 마일리지를 통해 price 함수로 결제할 값을 구하고 결제 확인을 받는다. 사용자가 카드 비밀번호를 입력하고 해당 비밀번호가 일치하면 예약 테이블에 새 테이블을 삽입한다. 만약 쿠폰을 사용했을 경우 쿠폰테이블에서 해당 테이블을 삭제한다. 그리고 만약 쿠폰 발행 조건(5번 추가 예약할 때마다 발행)을 충족했을 경우 새로운 쿠폰을 발행한다. 그 이후 사용한 마일리지와 얻은 마일리지를 바탕으로 최종 마일리지 업데이트를 진행한 후 home으로 돌아간다.

다음은 기능을 위한 함수에 대한 설명이다.

printCoupon 함수는 사용가능한 쿠폰을 출력하는 함수이다. 발급 받은 고객 ID와 할인금액 등 쿠폰정보가 기입된 쿠폰 목록을 출력하고 총 쿠폰 개수를 출력한다.

printFlight 함수는 편명을 출력하는 함수이다. 쿼리를 인자로 받아서 해당 쿼리에 해당하는 편명 정보를 보기 쉽게 출력한다.

couponExists 함수는 쿠폰 ID를 인자로 받아 해당 쿠폰이 존재하는지 여부를 리턴하는 함수이다.

getAvailableMileage 함수는 DB2024_Reservations 릴레이션을 사용하여 그 고객이 사용 가능한 마일리지가 얼마인지 계산하는 함수이다.

totalMilegeCalculate 함수는 DB2024_Reservations 릴레이션을 사용하여 그 고객의 총 마일리지가 얼마인지 계산하는 함수이다.

price 함수는 사용한 마일리지와 쿠폰을 고려하여 지불해야 하는 가격을 계산하는 함수이다.

newCoupon 함수는 쿠폰 발행 조건을 확인하는 함수이다. 사용한 쿠폰 수, 보유한 쿠폰 수, 예약 횟수를 얻어온 뒤 (사용한 쿠폰 수 + 쿠폰함에 보유하고 있는 쿠폰 수) * 5 + 4 이 예약횟수일때 쿠폰을 발급하도록 하였다.

retrieveCoupon 함수는 쿠폰 회수 조건을 확인하는 함수이다. 사용한 쿠폰 수, 보유한 쿠폰 수, 예약 횟수를 얻어온 뒤 (사용한 쿠폰 수 + 쿠폰함에 보유하고 있는 쿠폰 수) * 5 - 1이 예약 횟수일때 쿠폰을 삭제하도록 하였다.

printReserved는 고객 ID를 인자로 받아 DB2024_Reservations 릴레이션에서 그 고객의 예약을 찾아 읽기 쉽게 출력하는 함수이다.

main 함수에서는 Ewha Sky 대문을 꾸며주는 출력문을 넣고 바로 login 함수를 호출하도록 하였다.

(6) 17개의 요구사항

(1) 5개 이상의 테이블을 가지고 있어야 하고 각 테이블들의 컬럼(Attribute)의 수를 합하면 20개 이상이어야 한다.

-> 테이블은 DB2024_Customer, DB2024_Airplane, DB2024_FlightName, DB2024_Reservations, DB2024_Coupon로 총 5개이다.

DB2024_Customer은 custid, name, birthday, ID, PW, phoneNum, cardNum, payPW, usedCouponNum으로 총 9개의 컬럼을 가지고 있다.

DB2024_Airplane은 airlineName, airplaneID, capNum으로 총 3개의 컬럼을 가지고 있다.

DB2024_FlightName은 flightNum, goDate, timeT, startCountry, endCountry, price, mile, reservedNum, airplaneID으로 총 9개의 컬럼을 가지고 있다.

DB2024_Reservations는 reserveNum, isCouponUsed, useMile, earnMile, totalMile, payPrice, updateTime, state, custid, flightNum, goDate, 총 11개의 컬럼을 가지고 있다.

DB2024_Coupon은 couponID, couponDiscount, custid, 총 3개의 컬럼을 가지고 있다.

따라서 총 35개의 컬럼을 가지고 있다.

```
CREATE TABLE IF NOT EXISTS `DB2024Team09`.`DB2024_Customer` (
                `custid` INT NOT NULL,
                `name` VARCHAR(20) NOT NULL,
                `birthday` DATE NOT NULL,
                `ID` VARCHAR(20) NOT NULL,
                `PW` VARCHAR(20) NOT NULL,
                `phoneNum` CHAR(11) NULL,
                `cardNum` CHAR(14) NOT NULL,
                `payPW` CHAR(4) NOT NULL,
                `usedCouponNum` INT NOT NULL DEFAULT 0,
                PRIMARY KEY (`custid`))
ENGINE = InnoDB;
)
CREATE TABLE IF NOT EXISTS `DB2024Team09`.`DB2024_Airplane` (
`airlineName` VARCHAR(20) NOT NULL,
                `airplaneID` VARCHAR(45) NOT NULL,
                `capNum` INT NOT NULL,
                PRIMARY KEY (`airplaneID`))
```

```
ENGINE = InnoDB;
)
CREATE TABLE IF NOT EXISTS `DB2024Team09`.`DB2024_FlightName` (
                `flightNum` CHAR(5) NOT NULL,
                `goDate` DATETIME NOT NULL,
                `timeT` INT NOT NULL,
                `startCountry` VARCHAR(10) NOT NULL,
                `endCountry` VARCHAR(10) NOT NULL,
                `price` INT NOT NULL,
                `mile` INT NOT NULL,
                `reservedNum` INT NOT NULL,
                `DB2024_Airplane_airplaneID` VARCHAR(45) NOT NULL,
                PRIMARY KEY (`flightNum`, `goDate`),
INDEX `fk_DB2024_FlightName_DB2024_Airplane1_idx` (`DB2024_Airplane_airplaneID`
ASC) VISIBLE,
                CONSTRAINT `fk_DB2024_FlightName_DB2024_Airplane1`
                FOREIGN KEY (`DB2024_Airplane_airplaneID`)
                REFERENCES `DB2024Team09`.`DB2024_Airplane` (`airplaneID`)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION)
ENGINE = InnoDB
)
CREATE TABLE IF NOT EXISTS `DB2024Team09`.`DB2024_Reservations` (
                `reserveNum` INT NOT NULL AUTO_INCREMENT,
                `isCouponUsed` VARCHAR(5) NOT NULL,
                `useMile` INT NOT NULL,
                `earnMile` INT NOT NULL,
                `totalMile` INT NOT NULL,
                `payPrice` INT NOT NULL,
                `updateTime` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
                `state` VARCHAR(5) NOT NULL,
                `DB2024_Customer_custid` INT NOT NULL,
```

```
		`DB2024_FlightName_flightNum` CHAR(5) NOT NULL,
		`DB2024_FlightName_goDate` DATETIME NOT NULL,
 INDEX `fk_DB2024_Reservations_DB2024_Customer1_idx` (`DB2024_Customer_custid`
ASC) VISIBLE,
INDEX `fk_DB2024_Reservations_DB2024_FlightName1_idx`
(`DB2024_FlightName_flightNum` ASC, `DB2024_FlightName_goDate` ASC) VISIBLE,
		PRIMARY KEY (`reserveNum`),
		CONSTRAINT `fk_DB2024_Reservations_DB2024_Customer1`
		FOREIGN KEY (`DB2024_Customer_custid`)
		REFERENCES `DB2024Team09`.`DB2024_Customer` (`custid`)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION,
		CONSTRAINT `fk_DB2024_Reservations_DB2024_FlightName1`
FOREIGN KEY (`DB2024_FlightName_flightNum` , `DB2024_FlightName_goDate`)
REFERENCES `DB2024Team09`.`DB2024_FlightName` (`flightNum` , `goDate`)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION)
ENGINE = InnoDB;
)
CREATE TABLE IF NOT EXISTS `DB2024Team09`.`DB2024_Coupon` (
		`couponID` INT NOT NULL AUTO_INCREMENT,
		`couponDiscount` INT NOT NULL,
		`DB2024_Customer_custid` INT NOT NULL,
		PRIMARY KEY (`couponID`),
INDEX `fk_DB2024_Coupon_DB2024_Customer_idx` (`DB2024_Customer_custid` ASC)
VISIBLE,
		CONSTRAINT `fk_DB2024_Coupon_DB2024_Customer`
		FOREIGN KEY (`DB2024_Customer_custid`)
		REFERENCES `DB2024Team09`.`DB2024_Customer` (`custid`)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

)

(2) 초기화를 위해 적어도 30개의 레코드(투플)를 가지고 있어야 한다. (모든 테이블들의 레코 드 수의 총합이 30개 이상)
-> DB2024_Customer에 삽입하는 튜플 40개, DB2024_Airplane에 삽입하는 투플 6개, DB2024_FlightName에 삽입하는 튜플 180개로 총 226개이다.

INSERT INTO DB2024_Customer(custid, name, birthday, ID, Pw, phoneNum, cardNum, payPW)
VALUES
(1, '김이름', STR_TO_DATE('2000-01-01', '%Y-%m-%d'), 'myid', 'password', '01011111111', '12341234123412', '1111'),
(2, '하지연', STR_TO_DATE('2003-04-03', '%Y-%m-%d'), 'lina4544', '030403', '01022584544', '51326848699523', '8484'),
(3, '양인경', STR_TO_DATE('2001-11-30', '%Y-%m-%d'), 'highlighjjang', 'hl20091016', '01024579872', '12345678912345', '1234'),
(4, '안채연', STR_TO_DATE('2002-08-14', '%Y-%m-%d'), 'cheayeon02', 'acy0814', '01020458330', '69382692047532', '7489'),
(5, '이라희', STR_TO_DATE('2006-12-15', '%Y-%m-%d'), 'lhlove', 'asdfasdf', '01065657979', '95362268458797', '2647'),
(6, '윤정한', STR_TO_DATE('1995-10-04', '%Y-%m-%d'), 'angel1004', 'carat1', '01030672936', '11024976830204', '2200'),
(7, '최승철', STR_TO_DATE('1995-08-08', '%Y-%m-%d'), 'iluvsvt', '102030', '01012345678', '30465744266879', '3013'),
(8, '권순영', STR_TO_DATE('1996-06-15', '%Y-%m-%d'), 'horang22', 'glglgl22', '01062196603', '50492711648599', '5944'),
(9, '이준호', STR_TO_DATE('1983-06-09', '%Y-%m-%d'), 'junho83', 'junho1234', '01014725836', '24680135792468', '1478'),
(10, '최윤서', STR_TO_DATE('1995-02-10', '%Y-%m-%d'), 'yoonseo95', 'YS!password', '01075395182', '98765432198765', '7531'),
(11, '백승우', STR_TO_DATE('1990-04-12', '%Y-%m-%d'), 'seungwoo90', 'ilovemydog', '01098765432', '36925814703692', '9874'),

(12, '임지우', STR_TO_DATE('1997-08-02', '%Y-%m-%d'), ' jiwoo97', 'chocolate12',
'01064877559', '74185296301478', '3690'),
(13, '한서연', STR_TO_DATE('1992-05-11', '%Y-%m-%d'), 'seoyeon92', 'hsy_0511',
'01036925814', '36925814785236', '9632'),
(14, '이재현', STR_TO_DATE('1997-09-13','%Y-%m-%d'), 'leejaehyunow', 'wogus13',
'01012165859', '12886549320445', '9138'),
(15, '이정환', STR_TO_DATE('1992-03-20','%Y-%m-%d'), 'samdol2', 'mountain69',
'01042357318', '21556812280147', '3320'),
(16, '정소민', STR_TO_DATE('1965-07-26','%Y-%m-%d'), 'littlemin727', 'jsm0727',
'01019029338', '16579314680041', '6709'),
(17, '강민서', STR_TO_DATE('2003-02-26','%Y-%m-%d'), 'kinda03', 'kms2271006',
'01045337338', '26579315680024', '3225'),
(18, '김지원', STR_TO_DATE('2002-02-21', '%Y-%m-%d'), 'lovelyjiwon', 'jiwon0221',
'01023426780', '34412340940113', '0341'),
(19, '이지훈', STR_TO_DATE('1989-01-23', '%Y-%m-%d'), 'jihun82', 'p@ssw0rd',
'01012345678', '98765432101234', '5678'),
(20, '송지은', STR_TO_DATE('1975-08-27', '%Y-%m-%d'), 'jieun75', 'mySecret!',
'01098765432', '12345678901234', '1234'),
(21, '김주언', STR_TO_DATE('2000-01-01', '%Y-%m-%d'), 'kju123', 'pass1234',
'01012345678', '12341234123412', '1123'),
(22, '이철수', STR_TO_DATE('1995-03-15', '%Y-%m-%d'), 'leecs', 'password95',
'01023456789', '23452345234523', '1224'),
(23, '박영희', STR_TO_DATE('1988-07-20', '%Y-%m-%d'), 'parkyh', 'yeonhee88',
'01034567890', '34563456345634', '1325'),
(24, '정지석', STR_TO_DATE('1976-11-05', '%Y-%m-%d'), 'jungjs', 'jiseok76', '01045678901',
'45674567456745', '1426'),
(25, '홍길동', STR_TO_DATE('1990-09-10', '%Y-%m-%d'), 'honggd', 'gildong90',
'01056789012', '56785678567856', '1527'),
(26, '김영수', STR_TO_DATE('1983-05-25', '%Y-%m-%d'), 'kimys', 'yseosu83',
'01067890123', '67896789678967', '1628'),
(27, '이지민', STR_TO_DATE('1998-02-28', '%Y-%m-%d'), 'leejm', 'jimin98', '01078901234',
'78907890789078', '1729'),

(28, '박성호', STR_TO_DATE('1985-04-14', '%Y-%m-%d'), 'parksh', 'sung-ho85', '01089012345', '89018901890189', '1830'),
(29, '정유진', STR_TO_DATE('1992-12-30', '%Y-%m-%d'), 'jeongyj', 'yujin92', '01090123456', '90129012901290', '1931'),
(30, '홍성민', STR_TO_DATE('1979-08-17', '%Y-%m-%d'), 'hongs', 'sungmin79', '01061234567', '21230123012301', '2032'),
(31, '김민지', STR_TO_DATE('2001-06-03', '%Y-%m-%d'), 'kimminj', 'minji01', '01066957258', '15495875369587', '1133'),
(32, '이지원', STR_TO_DATE('1996-10-18', '%Y-%m-%d'), 'leejw', 'jiwon96', '01098745454', '23452345234523', '1234'),
(33, '박지우', STR_TO_DATE('1989-11-25', '%Y-%m-%d'), 'parkjw', 'jiwoo89', '01022222222', '34563456345634', '1335'),
(34, '정민석', STR_TO_DATE('1977-03-11', '%Y-%m-%d'), 'jeongms', 'minseok77', '01033333333', '45674567456745', '1436'),
(35, '홍수진', STR_TO_DATE('1991-01-22', '%Y-%m-%d'), 'hongsj', 'soojin91', '01044444444', '56785678567856', '1537'),
(36, '김현우', STR_TO_DATE('1984-07-09', '%Y-%m-%d'), 'kimhw', 'hyunwoo84', '01055555555', '67896789678967', '1638'),
(37, '이승현', STR_TO_DATE('1999-05-14', '%Y-%m-%d'), 'leesh', 'seunghyun99', '01066666666', '78907890789078', '1739'),
(38, '박민지', STR_TO_DATE('1986-09-27', '%Y-%m-%d'), 'parkmj', 'minji86', '01077777777', '89018901890189', '1830'),
(39, '이지수', STR_TO_DATE('1993-07-02', '%Y-%m-%d'), 'leejs', 'jisoo93', '01088888888', '90129012901290', '1931'),
(40, '김성우', STR_TO_DATE('1980-12-12', '%Y-%m-%d'), 'kimsu', 'sungwoo80', '01099999999', '81230123012301', '2032');

INSERT INTO DB2024_Airplane
VALUES
('대한항공', 'HL7790', 70),
('아시아나항공', 'HL7387', 50),
('아시아나항공', 'HL7324', 50),

('진에어', 'HL7100', 30),

('진에어', 'HL7101', 30),

('진에어', 'HL7102', 30);

INSERT INTO DB2024_FlightName(flightNum, DB2024_Airplane_airplaneID, goDate, timeT, startCountry, endCountry, reservedNum, price , mile)

VALUES

('KE260', 'HL7790', '2024-06-04 19:00:00', 14, '한국', '미국', 0, 1400000, 140000),

('KE480', 'HL7790', '2024-06-08 10:23:00', 2, '한국', '일본', 2, 200000, 20000),

('KE240', 'HL7790', '2024-06-08 18:13:00', 14, '한국', '미국', 0, 1400000, 140000),

('KE052', 'HL7790', '2024-06-09 22:08:00', 14, '미국', '한국', 5, 1400000, 140000),

('KE005', 'HL7790', '2024-06-10 15:58:00', 3, '중국', '한국', 0, 300000, 30000),

('KE250', 'HL7790', '2024-06-21 17:56:00', 14, '한국', '미국', 7, 1400000, 140000),

('KE095', 'HL7790', '2024-06-22 11:46:00', 3, '중국', '한국', 9, 300000, 30000),

('KE055', 'HL7790', '2024-06-25 21:30:00', 3, '중국', '한국', 0, 300000, 30000),

('KE510', 'HL7790', '2024-06-29 16:45:00', 3, '한국', '중국', 4, 300000, 30000),

('KE012', 'HL7790', '2024-07-02 13:30:00', 14, '미국', '한국', 0, 1400000, 140000),

('KE075', 'HL7790', '2024-07-05 20:58:00', 3, '중국', '한국', 0, 300000, 30000),

('KE045', 'HL7790', '2024-07-08 12:00:00', 3, '중국', '한국', 0, 300000, 30000),

('KE290', 'HL7790', '2024-07-08 12:44:00', 14, '한국', '미국', 1, 1400000, 140000),

('KE034', 'HL7790', '2024-07-08 15:38:00', 2, '일본', '한국', 0, 200000, 20000),

('KE510', 'HL7790', '2024-07-08 15:39:00', 3, '한국', '중국', 0, 300000, 30000),

('KE540', 'HL7790', '2024-07-12 13:16:00', 3, '한국', '중국', 0, 300000, 30000),

('KE470', 'HL7790', '2024-07-15 12:32:00', 2, '한국', '일본', 0, 200000, 20000),

('KE460', 'HL7790', '2024-07-15 22:42:00', 2, '한국', '일본', 0, 200000, 20000),

('KE025', 'HL7790', '2024-07-16 10:24:00', 3, '중국', '한국', 0, 300000, 30000),

('KE054', 'HL7790', '2024-07-17 12:45:00', 2, '일본', '한국', 5, 200000, 20000),

('KE460', 'HL7790', '2024-07-19 21:10:00', 2, '한국', '일본', 4, 200000, 20000),

('KE580', 'HL7790', '2024-07-21 16:29:00', 3, '한국', '중국', 0, 300000, 30000),

('KE450', 'HL7790', '2024-07-25 20:18:00', 2, '한국', '일본', 0, 200000, 20000),

('KE520', 'HL7790', '2024-08-05 13:19:00', 3, '한국', '중국', 4, 300000, 30000),

('KE084', 'HL7790', '2024-08-09 18:57:00', 2, '일본', '한국', 4, 200000, 20000),

('KE025', 'HL7790', '2024-08-14 12:21:00', 3, '중국', '한국', 0, 300000, 30000),
('KE530', 'HL7790', '2024-08-14 21:07:00', 3, '한국', '중국', 7, 300000, 30000),
('KE530', 'HL7790', '2024-08-20 21:01:00', 3, '한국', '중국', 0, 300000, 30000),
('KE004', 'HL7790', '2024-08-26 11:44:00', 2, '일본', '한국', 0, 200000, 20000),
('KE085', 'HL7790', '2024-08-26 13:31:00', 3, '중국', '한국', 0, 300000, 30000),

('OZ092', 'HL7387', '2024-06-08 10:20:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ075', 'HL7387', '2024-06-13 17:03:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ580', 'HL7387', '2024-06-15 16:59:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ290', 'HL7387', '2024-06-15 22:36:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ460', 'HL7387', '2024-06-17 08:49:00', 2, '한국', '일본', 0, 160000, 11200),
('OZ290', 'HL7387', '2024-06-19 11:07:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ044', 'HL7387', '2024-06-20 18:42:00', 2, '일본', '한국', 0, 160000, 11200),
('OZ560', 'HL7387', '2024-06-24 15:10:00', 3, '한국', '중국', 5, 240000, 16800),
('OZ002', 'HL7387', '2024-06-26 21:36:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ490', 'HL7387', '2024-06-28 09:08:00', 2, '한국', '일본', 0, 160000, 11200),
('OZ260', 'HL7387', '2024-07-03 22:34:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ460', 'HL7387', '2024-07-04 21:26:00', 2, '한국', '일본', 0, 160000, 11200),
('OZ085', 'HL7387', '2024-07-05 15:00:00', 3, '중국', '한국', 7, 240000, 16800),
('OZ450', 'HL7387', '2024-07-10 20:48:00', 2, '한국', '일본', 0, 160000, 11200),
('OZ085', 'HL7387', '2024-07-11 15:43:00', 3, '중국', '한국', 6, 240000, 16800),
('OZ230', 'HL7387', '2024-07-12 16:57:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ560', 'HL7387', '2024-07-14 15:46:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ490', 'HL7387', '2024-07-20 13:39:00', 2, '한국', '일본', 0, 160000, 11200),
('OZ034', 'HL7387', '2024-07-22 15:17:00', 2, '일본', '한국', 0, 160000, 11200),
('OZ430', 'HL7387', '2024-07-28 15:42:00', 2, '한국', '일본', 0, 160000, 11200),
('OZ012', 'HL7387', '2024-07-29 13:41:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ460', 'HL7387', '2024-07-30 19:41:00', 2, '한국', '일본', 5, 160000, 11200),
('OZ400', 'HL7387', '2024-08-03 20:24:00', 2, '한국', '일본', 0, 160000, 11200),
('OZ025', 'HL7387', '2024-08-10 18:27:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ520', 'HL7387', '2024-08-14 20:56:00', 3, '한국', '중국', 0, 240000, 16800),

('OZ094', 'HL7387', '2024-08-16 13:09:00', 2, '일본', '한국', 0, 160000, 11200),
('OZ510', 'HL7387', '2024-08-19 16:08:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ092', 'HL7387', '2024-08-19 21:56:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ560', 'HL7387', '2024-08-22 14:38:00', 3, '한국', '중국', 0, 240000, 16800),

('OZ430', 'HL7387', '2024-08-23 22:29:00', 2, '한국', '일본', 0, 160000, 11200),

('OZ002', 'HL7324', '2024-06-04 19:38:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ240', 'HL7324', '2024-06-06 18:43:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ400', 'HL7324', '2024-06-08 12:28:00', 2, '한국', '일본', 0, 160000, 11200),
('OZ540', 'HL7324', '2024-06-08 14:34:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ025', 'HL7324', '2024-06-17 08:39:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ460', 'HL7324', '2024-06-21 20:14:00', 2, '한국', '일본', 7, 160000, 11200),
('OZ042', 'HL7324', '2024-06-25 16:33:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ005', 'HL7324', '2024-06-30 10:53:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ072', 'HL7324', '2024-07-01 17:36:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ280', 'HL7324', '2024-07-02 10:45:00', 14, '한국', '미국', 8, 1120000, 78400),
('OZ042', 'HL7324', '2024-07-02 19:41:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ054', 'HL7324', '2024-07-06 12:27:00', 2, '일본', '한국', 0, 160000, 11200),
('OZ590', 'HL7324', '2024-07-08 11:47:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ082', 'HL7324', '2024-07-11 10:29:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ540', 'HL7324', '2024-07-13 11:49:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ560', 'HL7324', '2024-07-15 14:01:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ200', 'HL7324', '2024-07-18 13:33:00', 14, '한국', '미국', 1, 1120000, 78400),
('OZ052', 'HL7324', '2024-07-21 22:24:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ012', 'HL7324', '2024-07-24 10:28:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ032', 'HL7324', '2024-07-24 18:51:00', 14, '미국', '한국', 11, 1120000, 78400),
('OZ460', 'HL7324', '2024-07-26 12:06:00', 2, '한국', '일본', 0, 160000, 11200),
('OZ500', 'HL7324', '2024-08-03 16:13:00', 3, '한국', '중국', 11, 240000, 16800),
('OZ570', 'HL7324', '2024-08-12 14:17:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ002', 'HL7324', '2024-08-13 14:29:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ084', 'HL7324', '2024-08-13 16:10:00', 2, '일본', '한국', 0, 160000, 11200),

('OZ065', 'HL7324', '2024-08-13 17:12:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ095', 'HL7324', '2024-08-16 21:54:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ260', 'HL7324', '2024-08-17 12:35:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ082', 'HL7324', '2024-08-27 09:29:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ220', 'HL7324', '2024-08-27 17:55:00', 14, '한국', '미국', 0, 1120000, 78400),

('LJ550', 'HL7100', '2024-06-05 13:25:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ012', 'HL7100', '2024-06-06 08:11:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ034', 'HL7100', '2024-06-06 10:46:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ460', 'HL7100', '2024-06-14 10:52:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ024', 'HL7100', '2024-06-14 18:04:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ280', 'HL7100', '2024-06-15 19:25:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ085', 'HL7100', '2024-06-16 19:18:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ210', 'HL7100', '2024-06-18 09:26:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ520', 'HL7100', '2024-06-19 13:57:00', 3, '한국', '중국', 11, 150000, 7500),
('LJ085', 'HL7100', '2024-06-24 13:43:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ590', 'HL7100', '2024-06-29 21:26:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ094', 'HL7100', '2024-07-03 13:55:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ460', 'HL7100', '2024-07-06 17:38:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ560', 'HL7100', '2024-07-07 15:41:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ400', 'HL7100', '2024-07-15 19:53:00', 2, '한국', '일본', 11, 100000, 5000),
('LJ400', 'HL7100', '2024-07-20 16:27:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ470', 'HL7100', '2024-07-28 14:11:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ044', 'HL7100', '2024-08-01 11:18:00', 2, '일본', '한국', 11, 100000, 5000),
('LJ075', 'HL7100', '2024-08-01 14:09:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ002', 'HL7100', '2024-08-07 14:55:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ480', 'HL7100', '2024-08-10 22:53:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ092', 'HL7100', '2024-08-11 14:29:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ460', 'HL7100', '2024-08-11 19:43:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ530', 'HL7100', '2024-08-14 14:07:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ004', 'HL7100', '2024-08-17 19:40:00', 2, '일본', '한국', 11, 100000, 5000),

('LJ072', 'HL7100', '2024-08-22 12:08:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ082', 'HL7100', '2024-08-22 17:37:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ065', 'HL7100', '2024-08-26 08:34:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ270', 'HL7100', '2024-08-27 12:46:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ210', 'HL7100', '2024-08-30 17:16:00', 14, '한국', '미국', 0, 700000, 35000),

('LJ410', 'HL7101', '2024-06-04 16:43:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ470', 'HL7101', '2024-06-05 11:12:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ520', 'HL7101', '2024-06-06 20:59:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ022', 'HL7101', '2024-06-09 12:18:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ580', 'HL7101', '2024-06-12 22:03:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ004', 'HL7101', '2024-06-13 19:16:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ570', 'HL7101', '2024-06-15 15:02:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ042', 'HL7101', '2024-06-18 11:02:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ280', 'HL7101', '2024-06-28 18:57:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ210', 'HL7101', '2024-06-28 22:42:00', 14, '한국', '미국', 11, 700000, 35000),
('LJ074', 'HL7101', '2024-06-29 15:40:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ280', 'HL7101', '2024-07-02 11:16:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ044', 'HL7101', '2024-07-03 12:54:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ250', 'HL7101', '2024-07-04 22:30:00', 14, '한국', '미국', 11, 700000, 35000),
('LJ095', 'HL7101', '2024-07-07 11:16:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ520', 'HL7101', '2024-07-10 17:56:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ015', 'HL7101', '2024-07-12 18:34:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ055', 'HL7101', '2024-07-14 11:04:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ044', 'HL7101', '2024-07-14 15:03:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ052', 'HL7101', '2024-07-24 09:09:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ540', 'HL7101', '2024-07-24 13:27:00', 3, '한국', '중국', 11, 150000, 7500),
('LJ280', 'HL7101', '2024-08-01 12:40:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ240', 'HL7101', '2024-08-01 19:30:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ450', 'HL7101', '2024-08-04 16:15:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ400', 'HL7101', '2024-08-16 17:02:00', 2, '한국', '일본', 0, 100000, 5000),

('LJ015', 'HL7101', '2024-08-23 16:23:00', 3, '중국', '한국', 11, 150000, 7500),
('LJ470', 'HL7101', '2024-08-24 08:01:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ045', 'HL7101', '2024-08-26 10:43:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ500', 'HL7101', '2024-08-28 14:21:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ062', 'HL7101', '2024-08-30 20:10:00', 14, '미국', '한국', 11, 700000, 35000),

('LJ210', 'HL7102', '2024-06-01 14:08:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ075', 'HL7102', '2024-06-02 18:35:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ560', 'HL7102', '2024-06-05 18:22:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ042', 'HL7102', '2024-06-07 17:11:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ490', 'HL7102', '2024-06-08 13:51:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ210', 'HL7102', '2024-06-11 11:05:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ240', 'HL7102', '2024-06-12 15:23:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ012', 'HL7102', '2024-06-13 13:10:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ072', 'HL7102', '2024-06-21 22:01:00', 14, '미국', '한국', 11, 700000, 35000),
('LJ450', 'HL7102', '2024-07-01 19:26:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ430', 'HL7102', '2024-07-06 09:39:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ500', 'HL7102', '2024-07-06 15:11:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ280', 'HL7102', '2024-07-06 20:04:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ530', 'HL7102', '2024-07-09 09:26:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ012', 'HL7102', '2024-07-11 21:31:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ025', 'HL7102', '2024-07-16 18:41:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ250', 'HL7102', '2024-07-18 13:58:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ230', 'HL7102', '2024-07-18 16:08:00', 14, '한국', '미국', 11, 700000, 35000),
('LJ002', 'HL7102', '2024-07-20 09:00:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ072', 'HL7102', '2024-07-20 19:41:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ230', 'HL7102', '2024-07-21 15:31:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ072', 'HL7102', '2024-07-24 20:54:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ084', 'HL7102', '2024-07-25 21:47:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ430', 'HL7102', '2024-07-29 14:06:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ065', 'HL7102', '2024-07-30 08:13:00', 3, '중국', '한국', 0, 150000, 7500),

('LJ032', 'HL7102', '2024-08-02 09:07:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ045', 'HL7102', '2024-08-03 15:54:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ260', 'HL7102', '2024-08-03 20:35:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ290', 'HL7102', '2024-08-06 14:41:00', 14, '한국', '미국', 11, 700000, 35000),
('LJ450', 'HL7102', '2024-08-06 16:17:00', 2, '한국', '일본', 0, 100000, 5000),

('KE210', 'HL7790', '2024-08-20 18:36:00', 14, '한국', '미국', 0, 1400000, 140000),
('KE012', 'HL7790', '2024-08-27 15:09:00', 14, '미국', '한국', 0, 1400000, 140000),
('KE490', 'HL7790', '2024-08-28 18:01:00', 2, '한국', '일본', 0, 200000, 20000),
('KE540', 'HL7790', '2024-08-29 19:09:00', 3, '한국', '중국', 0, 300000, 30000),
('OZ012', 'HL7387', '2024-06-03 19:48:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ095', 'HL7387', '2024-06-07 14:53:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ025', 'HL7387', '2024-06-09 16:00:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ230', 'HL7387', '2024-06-10 16:39:00', 14, '한국', '미국', 11, 1120000, 78400),
('OZ094', 'HL7387', '2024-06-17 21:45:00', 2, '일본', '한국', 0, 160000, 11200),
('OZ240', 'HL7387', '2024-06-25 20:47:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ530', 'HL7387', '2024-06-25 21:38:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ280', 'HL7387', '2024-06-29 12:37:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ290', 'HL7387', '2024-07-02 08:06:00', 14, '한국', '미국', 11, 1120000, 78400),
('OZ250', 'HL7387', '2024-07-02 21:58:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ280', 'HL7387', '2024-07-04 12:32:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ044', 'HL7387', '2024-07-06 09:18:00', 2, '일본', '한국', 0, 160000, 11200),
('OZ022', 'HL7387', '2024-07-06 14:02:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ260', 'HL7387', '2024-07-08 17:58:00', 14, '한국', '미국', 0, 1120000, 78400),

('OZ520', 'HL7387', '2024-07-10 20:26:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ024', 'HL7387', '2024-07-11 13:31:00', 2, '일본', '한국', 0, 160000, 11200),
('OZ092', 'HL7387', '2024-07-11 18:46:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ005', 'HL7387', '2024-07-17 16:35:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ014', 'HL7387', '2024-07-23 20:40:00', 2, '일본', '한국', 15, 160000, 11200),
('OZ092', 'HL7387', '2024-07-24 11:17:00', 14, '미국', '한국', 0, 1120000, 78400),

('OZ220', 'HL7387', '2024-07-24 17:31:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ470', 'HL7387', '2024-07-24 22:34:00', 2, '한국', '일본', 0, 160000, 11200),
('OZ510', 'HL7387', '2024-07-28 21:08:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ025', 'HL7387', '2024-07-30 10:29:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ072', 'HL7387', '2024-08-06 22:34:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ280', 'HL7387', '2024-08-19 14:03:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ410', 'HL7387', '2024-08-21 19:10:00', 2, '한국', '일본', 0, 160000, 11200),
('OZ054', 'HL7387', '2024-08-22 08:52:00', 2, '일본', '한국', 0, 160000, 11200),
('OZ075', 'HL7387', '2024-08-26 17:57:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ240', 'HL7324', '2024-06-04 15:12:00', 14, '한국', '미국', 4, 1120000, 78400),
('OZ094', 'HL7324', '2024-06-08 22:39:00', 2, '일본', '한국', 0, 160000, 11200),
('OZ540', 'HL7324', '2024-06-10 09:04:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ470', 'HL7324', '2024-06-11 20:04:00', 2, '한국', '일본', 0, 160000, 11200),
('OZ540', 'HL7324', '2024-06-18 13:56:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ230', 'HL7324', '2024-06-20 21:36:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ065', 'HL7324', '2024-06-23 12:57:00', 3, '중국', '한국', 9, 240000, 16800),
('OZ055', 'HL7324', '2024-07-02 11:28:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ280', 'HL7324', '2024-07-05 12:33:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ002', 'HL7324', '2024-07-06 19:07:00', 14, '미국', '한국', 6, 1120000, 78400),
('OZ075', 'HL7324', '2024-07-08 21:42:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ012', 'HL7324', '2024-07-09 18:05:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ580', 'HL7324', '2024-07-11 08:23:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ210', 'HL7324', '2024-07-17 09:03:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ052', 'HL7324', '2024-07-17 20:52:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ250', 'HL7324', '2024-07-17 21:57:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ500', 'HL7324', '2024-07-22 19:15:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ460', 'HL7324', '2024-08-02 21:35:00', 2, '한국', '일본', 0, 160000, 11200),
('OZ014', 'HL7324', '2024-08-03 09:18:00', 2, '일본', '한국', 0, 160000, 11200),
('OZ580', 'HL7324', '2024-08-11 17:47:00', 3, '한국', '중국', 0, 240000, 16800),
('OZ095', 'HL7324', '2024-08-11 21:30:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ580', 'HL7324', '2024-08-12 10:35:00', 3, '한국', '중국', 8, 240000, 16800),
('OZ084', 'HL7324', '2024-08-14 11:33:00', 2, '일본', '한국', 0, 160000, 11200),

('OZ270', 'HL7324', '2024-08-14 22:52:00', 14, '한국', '미국', 0, 1120000, 78400),
('OZ095', 'HL7324', '2024-08-15 11:26:00', 3, '중국', '한국', 0, 240000, 16800),
('OZ074', 'HL7324', '2024-08-15 12:57:00', 2, '일본', '한국', 0, 160000, 11200),
('OZ032', 'HL7324', '2024-08-17 15:24:00', 14, '미국', '한국', 6, 1120000, 78400),
('OZ022', 'HL7324', '2024-08-17 22:59:00', 14, '미국', '한국', 0, 1120000, 78400),
('OZ270', 'HL7324', '2024-08-23 09:17:00', 14, '한국', '미국', 15, 1120000, 78400),
('OZ530', 'HL7324', '2024-08-26 17:38:00', 3, '한국', '중국', 0, 240000, 16800),
('LJ034', 'HL7100', '2024-06-02 19:53:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ094', 'HL7100', '2024-06-05 10:08:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ580', 'HL7100', '2024-06-05 15:29:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ092', 'HL7100', '2024-06-06 14:10:00', 14, '미국', '한국', 21, 700000, 35000),
('LJ560', 'HL7100', '2024-06-07 15:09:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ400', 'HL7100', '2024-06-12 19:49:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ082', 'HL7100', '2024-06-12 21:54:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ055', 'HL7100', '2024-06-15 19:53:00', 3, '중국', '한국', 7, 150000, 7500),
('LJ094', 'HL7100', '2024-06-19 08:23:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ470', 'HL7100', '2024-06-22 16:00:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ064', 'HL7100', '2024-06-24 08:24:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ064', 'HL7100', '2024-06-30 13:37:00', 2, '일본', '한국', 6, 100000, 5000),
('LJ400', 'HL7100', '2024-07-03 10:34:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ004', 'HL7100', '2024-07-05 17:50:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ022', 'HL7100', '2024-07-08 12:51:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ035', 'HL7100', '2024-07-10 08:31:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ240', 'HL7100', '2024-07-15 11:11:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ280', 'HL7100', '2024-07-15 18:51:00', 14, '한국', '미국', 6, 700000, 35000),
('LJ095', 'HL7100', '2024-07-26 19:23:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ250', 'HL7100', '2024-07-28 11:16:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ290', 'HL7100', '2024-07-29 16:08:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ520', 'HL7100', '2024-07-29 20:16:00', 3, '한국', '중국', 6, 150000, 7500),
('LJ045', 'HL7100', '2024-08-06 12:36:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ044', 'HL7100', '2024-08-09 18:41:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ065', 'HL7100', '2024-08-10 09:18:00', 3, '중국', '한국', 0, 150000, 7500),

('LJ025', 'HL7100', '2024-08-10 21:52:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ005', 'HL7100', '2024-08-17 08:04:00', 3, '중국', '한국', 14, 150000, 7500),
('LJ035', 'HL7100', '2024-08-18 19:07:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ064', 'HL7100', '2024-08-25 14:14:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ270', 'HL7100', '2024-08-25 15:31:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ002', 'HL7101', '2024-06-05 17:39:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ004', 'HL7101', '2024-06-05 18:28:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ012', 'HL7101', '2024-06-07 18:51:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ450', 'HL7101', '2024-06-07 19:59:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ045', 'HL7101', '2024-06-10 17:55:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ230', 'HL7101', '2024-06-11 16:15:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ410', 'HL7101', '2024-06-12 16:34:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ420', 'HL7101', '2024-06-16 20:31:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ440', 'HL7101', '2024-06-22 10:45:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ460', 'HL7101', '2024-06-24 10:19:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ084', 'HL7101', '2024-06-24 19:16:00', 2, '일본', '한국', 9, 100000, 5000),
('LJ530', 'HL7101', '2024-07-01 15:18:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ200', 'HL7101', '2024-07-02 16:21:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ570', 'HL7101', '2024-07-03 15:19:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ440', 'HL7101', '2024-07-05 12:59:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ540', 'HL7101', '2024-07-08 22:14:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ270', 'HL7101', '2024-07-11 18:01:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ095', 'HL7101', '2024-07-12 20:00:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ072', 'HL7101', '2024-07-12 20:01:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ024', 'HL7101', '2024-07-20 09:46:00', 2, '일본', '한국', 12, 100000, 5000),
('LJ280', 'HL7101', '2024-07-26 16:40:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ025', 'HL7101', '2024-08-02 18:02:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ490', 'HL7101', '2024-08-03 17:27:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ014', 'HL7101', '2024-08-07 21:24:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ420', 'HL7101', '2024-08-13 22:59:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ052', 'HL7101', '2024-08-15 14:11:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ092', 'HL7101', '2024-08-22 20:56:00', 14, '미국', '한국', 0, 700000, 35000),

('LJ064', 'HL7101', '2024-08-24 17:21:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ085', 'HL7101', '2024-08-24 18:01:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ095', 'HL7101', '2024-08-25 16:17:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ065', 'HL7102', '2024-06-06 21:31:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ005', 'HL7102', '2024-06-07 08:36:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ220', 'HL7102', '2024-06-07 22:52:00', 14, '한국', '미국', 7, 700000, 35000),
('LJ005', 'HL7102', '2024-06-11 11:17:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ014', 'HL7102', '2024-06-13 17:01:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ062', 'HL7102', '2024-06-15 21:29:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ054', 'HL7102', '2024-06-19 16:32:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ035', 'HL7102', '2024-06-19 19:17:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ230', 'HL7102', '2024-06-22 17:53:00', 14, '한국', '미국', 6, 700000, 35000),
('LJ065', 'HL7102', '2024-06-26 14:08:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ004', 'HL7102', '2024-06-28 22:03:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ240', 'HL7102', '2024-06-29 16:42:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ045', 'HL7102', '2024-06-29 17:10:00', 3, '중국', '한국', 0, 150000, 7500),
('LJ074', 'HL7102', '2024-07-04 18:54:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ200', 'HL7102', '2024-07-06 08:29:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ014', 'HL7102', '2024-07-07 11:01:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ280', 'HL7102', '2024-07-08 16:51:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ200', 'HL7102', '2024-07-17 15:00:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ450', 'HL7102', '2024-07-25 15:35:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ064', 'HL7102', '2024-07-25 16:56:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ430', 'HL7102', '2024-07-26 21:37:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ072', 'HL7102', '2024-08-03 14:06:00', 14, '미국', '한국', 0, 700000, 35000),
('LJ084', 'HL7102', '2024-08-09 13:56:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ460', 'HL7102', '2024-08-10 12:00:00', 2, '한국', '일본', 0, 100000, 5000),
('LJ520', 'HL7102', '2024-08-13 22:06:00', 3, '한국', '중국', 0, 150000, 7500),
('LJ290', 'HL7102', '2024-08-15 16:13:00', 14, '한국', '미국', 0, 700000, 35000),
('LJ024', 'HL7102', '2024-08-17 15:55:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ084', 'HL7102', '2024-08-18 21:36:00', 2, '일본', '한국', 0, 100000, 5000),
('LJ024', 'HL7102', '2024-08-20 12:48:00', 2, '일본', '한국', 0, 100000, 5000),

('LJ560', 'HL7102', '2024-08-20 14:05:00', 3, '한국', '중국', 30, 150000, 7500);

(3) 기본 키(primary key), 외래 키(foreign key), not null 제약조건(not null constraints)를 포함해야 한다

-> DB2024_Customer의 Primary key는 custid이고, Foreign key는 존재하지 않으며 모든 속성이 NOT NULL이다.

DB2024_Airplane의 Primary key는 airplaneID이고, Foreign key는 존재하지 않으며 모든 속성이 NOT NULL이다.

DB2024_FlightName의 Primary key는 flightNum과 goDate이며, Foreign key는 airplaneID이고, 모든 속성이 NOT NULL이다.

DB2024_Reservation의 Primary key는 reserveNum이고, Foreign key는 custid, flightNum, goDate이고, 모든 속성이 NOT NULL이다.

DB2024_Coupon의 Primary key는 couponID이고, Foreign key는 custid이고, 모든 속성이 NOT NULL이다.

```
CREATE TABLE IF NOT EXISTS `DB2024Team09`.`DB2024_Customer` (
                `custid` INT NOT NULL,
                `name` VARCHAR(20) NOT NULL,
                `birthday` DATE NOT NULL,
                `ID` VARCHAR(20) NOT NULL,
                `PW` VARCHAR(20) NOT NULL,
                `phoneNum` CHAR(11) NULL,
                `cardNum` CHAR(14) NOT NULL,
                `payPW` CHAR(4) NOT NULL,
                `usedCouponNum` INT NOT NULL DEFAULT 0,
                PRIMARY KEY (`custid`))
ENGINE = InnoDB;
)
CREATE TABLE IF NOT EXISTS `DB2024Team09`.`DB2024_Airplane` (
`airlineName` VARCHAR(20) NOT NULL,
                `airplaneID` VARCHAR(45) NOT NULL,
                `capNum` INT NOT NULL,
```

```
                PRIMARY KEY (`airplaneID`))
ENGINE = InnoDB;
)
CREATE TABLE IF NOT EXISTS `DB2024Team09`.`DB2024_FlightName` (
                `flightNum` CHAR(5) NOT NULL,
                `goDate` DATETIME NOT NULL,
                `timeT` INT NOT NULL,
                `startCountry` VARCHAR(10) NOT NULL,
                `endCountry` VARCHAR(10) NOT NULL,
                `price` INT NOT NULL,
                `mile` INT NOT NULL,
                `reservedNum` INT NOT NULL,
                `DB2024_Airplane_airplaneID` VARCHAR(45) NOT NULL,
                PRIMARY KEY (`flightNum`, `goDate`),
INDEX `fk_DB2024_FlightName_DB2024_Airplane1_idx` (`DB2024_Airplane_airplaneID`
ASC) VISIBLE,
                CONSTRAINT `fk_DB2024_FlightName_DB2024_Airplane1`
                FOREIGN KEY (`DB2024_Airplane_airplaneID`)
                REFERENCES `DB2024Team09`.`DB2024_Airplane` (`airplaneID`)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION)
ENGINE = InnoDB
)
CREATE TABLE IF NOT EXISTS `DB2024Team09`.`DB2024_Reservations` (
                `reserveNum` INT NOT NULL AUTO_INCREMENT,
                `isCouponUsed` VARCHAR(5) NOT NULL,
                `useMile` INT NOT NULL,
                `earnMile` INT NOT NULL,
                `totalMile` INT NOT NULL,
                `payPrice` INT NOT NULL,
                `updateTime` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
                `state` VARCHAR(5) NOT NULL,
```

```
		`DB2024_Customer_custid` INT NOT NULL,
		`DB2024_FlightName_flightNum` CHAR(5) NOT NULL,
		`DB2024_FlightName_goDate` DATETIME NOT NULL,
 INDEX `fk_DB2024_Reservations_DB2024_Customer1_idx` (`DB2024_Customer_custid` ASC) VISIBLE,
INDEX `fk_DB2024_Reservations_DB2024_FlightName1_idx` (`DB2024_FlightName_flightNum` ASC, `DB2024_FlightName_goDate` ASC) VISIBLE,
		PRIMARY KEY (`reserveNum`),
		CONSTRAINT `fk_DB2024_Reservations_DB2024_Customer1`
		FOREIGN KEY (`DB2024_Customer_custid`)
		REFERENCES `DB2024Team09`.`DB2024_Customer` (`custid`)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION,
		CONSTRAINT `fk_DB2024_Reservations_DB2024_FlightName1`
FOREIGN KEY (`DB2024_FlightName_flightNum` , `DB2024_FlightName_goDate`)
REFERENCES `DB2024Team09`.`DB2024_FlightName` (`flightNum` , `goDate`)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION)
ENGINE = InnoDB;
)
CREATE TABLE IF NOT EXISTS `DB2024Team09`.`DB2024_Coupon` (
		`couponID` INT NOT NULL AUTO_INCREMENT,
		`couponDiscount` INT NOT NULL,
		`DB2024_Customer_custid` INT NOT NULL,
		PRIMARY KEY (`couponID`),
INDEX `fk_DB2024_Coupon_DB2024_Customer_idx` (`DB2024_Customer_custid` ASC) VISIBLE,
		CONSTRAINT `fk_DB2024_Coupon_DB2024_Customer`
		FOREIGN KEY (`DB2024_Customer_custid`)
		REFERENCES `DB2024Team09`.`DB2024_Customer` (`custid`)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;
)
```

(4) 적어도 2개의 뷰를 정의해야 한다.

```
-- 예약 내역 확인하는 뷰 생성
CREATE OR REPLACE VIEW DB2024_vw_reserved(custid, reserveNum, iscouponused,
payprice, state, flightNum, godate, endcountry, startcountry)
AS
SELECT r.DB2024_Customer_custid, r.reserveNum, r.isCouponUsed, r.payPrice, r.state,
f.flightNum, f.goDate, f.startCountry, f.endCountry
FROM DB2024_Reservations r
JOIN DB2024_FlightName f
ON r.db2024_flightname_flightNum = f.flightNum AND r.db2024_flightname_goDate =
f.goDate;
```

-> DB2024_vw_reserved 뷰는 DB2024_Reservations 테이블과 DB2024_FlightName 테이블의 데이터를 통합하여, 고객이 예약한 항공편에 대한 모든 관련 정보를 쉽게 조회할 수 있게 해줍니다.

```
-- 항공편들을 출발 시간 순으로 조회하는 뷰 생성
CREATE OR REPLACE VIEW DB2024_vw_orderByTime (flightNum, goDate, timeT,
startCountry, endCountry, price, mile, reservedNum, capNum)
AS
SELECT f.flightNum, f.goDate, f.timeT, f.startCountry, f.endCountry, f.price, f.mile,
f.reservedNum, a.capNum
FROM DB2024_FlightName f
JOIN DB2024_Airplane a ON f.DB2024_Airplane_airplaneID = a.airplaneID
ORDER BY f.goDate;
```

-> DB2024_vw_orderByTime 뷰는 DB2024_FlightName 테이블과 DB2024_Airplane 테이블의 데이터를 통합하여, 항공편 정보와 항공기 정보를 하나의 통합된 뷰로 제공합니다. 이 뷰는 출발 날짜 기준으로 정렬되어 있어, 사용자가 항공편을 시간순으로 쉽게 조회할 수 있도록 도와줍니다.

-- 항공편들을 가격 순으로 조회하는 뷰 생성

CREATE OR REPLACE VIEW DB2024_vw_orderByPrice (flightNum, goDate, timeT, startCountry, endCountry, price, mile, reservedNum, capNum)

AS

SELECT f.flightNum, f.goDate, f.timeT, f.startCountry, f.endCountry, f.price, f.mile, f.reservedNum, a.capNum

FROM DB2024_FlightName f

JOIN DB2024_Airplane a ON f.DB2024_Airplane_airplaneID = a.airplaneID

ORDER BY f.price;

-> DB2024_vw_orderByPrice 뷰는 DB2024_FlightName 테이블과 DB2024_Airplane 테이블의 데이터를 통합하여, 항공편 정보와 항공기 정보를 하나의 통합된 뷰로 제공합니다. 이 뷰는 항공편을 가격순으로 정렬하여 제공함으로써, 고객이 가장 저렴한 항공편을 쉽게 찾고 비교할 수 있도록 도와줍니다.

(5) 모든 테이블과 뷰의 이름들은 “DB2024_”라는 접두어를 가지고 있어야 한다.

고객 테이블 : Table `DB2024Team09`.`DB2024_Customer`

항공기 테이블 : Table `DB2024Team09`.`DB2024_Airplane`

편명 테이블 : Table `DB2024Team09`.`DB2024_FlightName`

예약 테이블 : Table `DB2024Team09`.`DB2024_Reservations`

쿠폰 테이블 : Table `DB2024Team09`.`DB2024_Coupon`

예약내역 뷰 : View DB2024_vw_reserved

시간순 뷰 : View DB2024_vw_orderByTime

가격순 뷰 : View DB2024_vw_orderByPrice

(6) 적어도 4개의 인덱스를 정의해야 한다.

-- 편명 인덱스 생성

CREATE INDEX ix1_DB2024_FlightName ON DB2024_FlightName(flightNum);

CREATE INDEX ix2_DB2024_FlightName ON DB2024_FlightName(godate);

-- 편명 인덱스 생성

INDEX `fk_DB2024_FlightName_DB2024_Airplane1_idx` (`DB2024_Airplane_airplaneID` ASC) VISIBLE

-- 예약 인덱스 생성

INDEX `fk_DB2024_Reservations_DB2024_Customer1_idx` (`DB2024_Customer_custid` ASC) VISIBLE,
INDEX `fk_DB2024_Reservations_DB2024_FlightName1_idx` (`DB2024_FlightName_flightNum` ASC, `DB2024_FlightName_goDate` ASC) VISIBLE

(7) 인덱스를 사용하는 쿼리들을 포함해야 한다.

```
String selectExistPayedQuery = "SELECT count(*) FROM db2024_reservations r WHERE r.DB2024_FlightName_flightNum = ? and date_format(r.DB2024_FlightName_goDate, '%Y-%m-%d %H:%i') =  ? and r.state = '결제 완료'";
String selectExistCanceledQuery = "SELECT count(*) FROM db2024_reservations r WHERE r.DB2024_FlightName_flightNum = ? and date_format(r.DB2024_FlightName_goDate, '%Y-%m-%d %H:%i') =  ? and r.state = '결제 취소'";
    String capNumQuery = "SELECT COUNT(*) FROM db2024_FlightName f, db2024_Airplane a\r\n"
        + "        where  f.flightNum = ? and date_format(f.goDate, '%Y-%m-%d %H:%i') = ?\r\n"
        + "        and f.reservedNum = a.capNum and f.DB2024_Airplane_airplaneID = a.airplaneID";
```

(8) 뷰를 사용하는 쿼리를 포함해야 한다.

1) 편명 검색결과 정렬에 맞춰 출력(시간순, 가격순)

```
// 시간순
        String query1 = "SELECT * FROM DB2024_vw_orderByTime \r\n"
                                + "WHERE goDate >= '" + startDay + "' AND goDate <= '" +
endDay + "' AND startcountry = '" + startCountry + "' AND endcountry = '" + endCountry + "'"
;
        // 가격순
        String query2 = "SELECT * FROM DB2024_vw_orderByPrice \r\n"
                                + "WHERE goDate >= '" + startDay + "' AND goDate <= '" +
endDay + "' AND startcountry = '" + startCountry + "' AND endcountry = '" + endCountry + "'"
;
```

2) 예약 내역 출력 쿼리

```
        String query1 = "SELECT *\r\n" + "FROM db2024_vw_reserved WHERE custid = " +
custid;
```

(9) 트랜잭션(transaction)을 포함해야 한다.

```
try {
            // 업데이트 부분 트랜잭션 시작
            conn.setAutoCommit(false);
            // 최종 결제 취소할 때 실행할 업데이트문
String updateReservationQuery = "UPDATE db2024_reservations\r\n" + "SET useMile =
0,\r\n"
                + "earnMile = 0,\r\n" + "totalMile = ?,\r\n" + "payPrice = 0,\r\n" + "state = '결제
취소'\r\n"
                + "WHERE reserveNum = " + cancel;

            // 취소할 예약 번호의 사용한 마일리지 쿼리
String usedMileInCancelReserveQuery = "SELECT useMile FROM db2024_reservations
WHERE reserveNum = "
                + cancel;
```

```
ResultSet usedMileInCancelResultSet =
stmt.executeQuery(usedMileInCancelReserveQuery);
                int usedMile = 0;
                while (usedMileInCancelResultSet.next()) {
                        usedMile = usedMileInCancelResultSet.getInt(1);
                }

            // 취소할 예약 번호의 적립한 마일리지 쿼리
String earnedMileInCancelReserveQuery = "SELECT earnMile FROM db2024_reservations
WHERE reserveNum = "
                + cancel;
ResultSet earnedMileInCancelResultSet =
stmt.executeQuery(earnedMileInCancelReserveQuery);
                int earnedMile = 0;
                while (earnedMileInCancelResultSet.next()) {
                        earnedMile = earnedMileInCancelResultSet.getInt(1);
                }

            // 환불되는 금액 쿼리, 업데이트에서는 사용 불필요하지만 출력을 위해 구함.
String payedPriceInCancelReserveQuery = "SELECT payPrice FROM db2024_reservations
WHERE reserveNum = "
                + cancel;
ResultSet payedPriceInCancelReserveResultSet =
stmt.executeQuery(payedPriceInCancelReserveQuery);
                int payedPrice = 0;
                while (payedPriceInCancelReserveResultSet.next()) {
                        payedPrice = payedPriceInCancelReserveResultSet.getInt(1);
                }
                System.out.println("일주일 내로 계좌로 " + payedPrice + "원이 환불
처리됩니다. ");
            // 지금까지의 누적 마일리지 쿼리
```

```
String totalMileBeforeCancelReserveQuery = "SELECT totalMile FROM
db2024_reservations \r\n"
                    + "WHERE updateTime = (SELECT MAX(updateTime) FROM
db2024_reservations)";
                    int totalMile = 0;
ResultSet totalMileBeforCancelResultSet =
stmt.executeQuery(totalMileBeforeCancelReserveQuery);
                    while (totalMileBeforCancelResultSet.next()) {
                            totalMile = totalMileBeforCancelResultSet.getInt(1);
                    }

                    int newTotalMile = totalMile - earnedMile + usedMile;
                // 마일리지 복구
PreparedStatement mUpdateInCancelReservepstmt =
conn.prepareStatement(updateReservationQuery);
                    mUpdateInCancelReservepstmt.setString(1, newTotalMile + "");
                    mUpdateInCancelReservepstmt.executeUpdate();

                // 쿠폰 회수 여부 결정
                // 쿠폰 회수한다면 coupon 테이블에서 최근에 발급된 쿠폰 회수
                    if (retrieveCoupon()) {
                  // 쿠폰 테이블에서 삭제 쿼리
                    String deleteCouponQuery = "DELETE FROM db2024_coupon \r\n"
                      + "WHERE \r\n"
+ "couponID = (select min from (select min(couponID) as min from db2024_coupon) as
c_t);";
                    stmt.executeUpdate(deleteCouponQuery);
                    System.out.println("!!예약 취소로 인해 가장 오래된 쿠폰이 삭제되었습니다!!");
                  }
                }
                catch (SQLException se) {
                  // 예약 취소 트랜잭션에서 오류 발생 시 쿠폰, 마일리지, 예약 내역 업데이트 롤백
```

```
        //se.printStackTrace();

        try{
          if(conn!=null)
            conn.rollback();
        }catch(SQLException se2){
          se2.printStackTrace();
          System.out.println("서버와 연결에 실패했습니다. ");
        }
      }
```

(10) 중첩된 쿼리(nested query)들을 가지는 쿼리들을 포함해야 한다.

```
String totalMileBeforeCancelReserveQuery = "SELECT totalMile FROM
db2024_reservations \r\n"+ "WHERE updateTime = (SELECT MAX(updateTime) FROM
db2024_reservations)";
String deleteCouponQuery = "DELETE FROM db2024_coupon \r\n"+ "WHERE \r\n"+
"couponID = (select min from (select min(couponID) as min from db2024_coupon) as c_t);";
String query = "SELECT r.totalMile FROM db2024_reservations r\r\n"+ "WHERE
r.reserveNum = (SELECT MAX(r2.reserveNum) FROM db2024_reservations r2) AND
DB2024_Customer_custid = "+ custid;
String mileQuery = "SELECT totalMile FROM DB2024_Reservations WHERE
DB2024_Customer_custid = " + custid+ "\r\n" + "AND reserveNum = (SELECT
MAX(reserveNum) FROM DB2024_Reservations)";
```

(11) 조인 쿼리(join query)들을 가지는 쿼리들은 포함해야 한다.

```
CREATE OR REPLACE VIEW DB2024_vw_reserved(custid, reserveNum, iscouponused,
payprice, state, flightNum, godate, endcountry, startcountry)
AS   SELECT r.DB2024_Customer_custid, r.reserveNum, r.isCouponUsed, r.payPrice,
r.state, f.flightNum, f.goDate 시간, f.startCountry, f.endCountry
                        FROM DB2024_Reservations r
```

JOIN DB2024_FlightName f

ON r.db2024_flightname_flightNum=f.flightNum AND

r.db2024_flightname_goDate=f.goDate;

-> 예약 내역 확인하는 뷰 생성

CREATE OR REPLACE VIEW DB2024_vw_orderByTime (flightNum, goDate, timeT, startCountry, endCountry, price, mile, reservedNum, capNum)

AS SELECT f.flightNum, f.goDate 시간 , f.timeT, f.startCountry, f.endCountry, f.price, f.mile, f.reservedNum, a.capNum

FROM DB2024_FlightName f JOIN DB2024_Airplane a ON f.DB2024_Airplane_airplaneID = a.airplaneID

ORDER BY f.goDate;

-> 항공 편들을 출발 시간 순으로 조회하는 뷰 생성

CREATE OR REPLACE VIEW DB2024_vw_orderByPrice (flightNum, goDate, timeT, startCountry, endCountry, price, mile, reservedNum, capNum)

AS SELECT f.flightNum, f.goDate 시간 , f.timeT, f.startCountry, f.endCountry, f.price, f.mile, f.reservedNum, a.capNum

FROM DB2024_FlightName f JOIN DB2024_Airplane a ON f.DB2024_Airplane_airplaneID = a.airplaneID

ORDER BY f.price;

-> 항공 편들을 가격 순으로 조회하는 뷰 생성

SELECT COUNT(*)

FROM db2024_FlightName f, db2024_Airplane a

where f.flightNum = ? and date_format(f.goDate, '%Y-%m-%d %H:%i') = ?

and f.reservedNum = a.capNum and f.DB2024_Airplane_airplaneID = a.airplaneID";

-> 예약인원이 수용인원에 도달한 편명 수 세기

(12) 매개 변수를 가지면서 동적으로 만드는 쿼리를 포함해야 한다. 다시 말해, 사용자로부터 입력 값을 받고 사용자가 입력한 값으로 쿼리를 생성한다.

다음은 사용자가 원하는 예약을 찾기 위해 검색하는 쿼리이다.

System.out.println("출발지를 입력하세요 : ");

String startCountry = s.next();

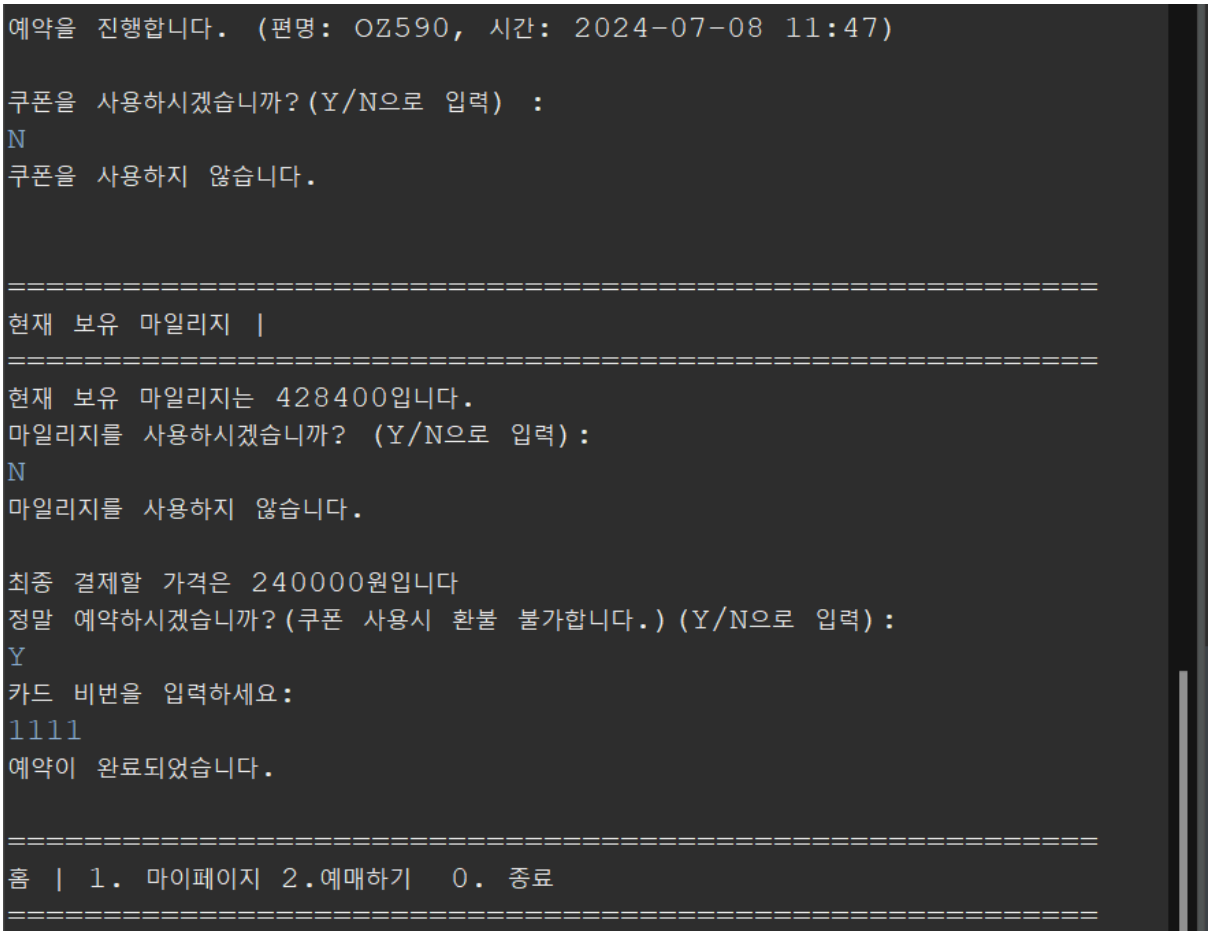
System.out.println("도착지를 입력하세요 : ");

```
String endCountry = s.next();
System.out.println("출발날짜를 나누어 차례로 입력하세요.");
System.out.println("출발날짜의 연도를 입력하세요 (ex. 2024) : ");
int year = s.nextInt();
System.out.println("출발날짜의 달(1~12)을 입력하세요 : ");
int month = s.nextInt();
System.out.println("출발날짜의 날짜(0~31)를 입력하세요 : ");
int day = s.nextInt();
System.out.println("출발날짜의 시(0~23)를 입력하세요 : ");
int hour = s.nextInt();
String query1 = "SELECT * FROM DB2024_vw_orderByTime \r\n"
                + "WHERE goDate >= '" + startDay + "' AND goDate <= '" + endDay + "' AND
startcountry = '" + startCountry + "' AND endcountry = '" + endCountry + "'" ;
```

(13) 그래픽 또는 문자 기반의 사용자 인터페이스를 사용해야 한다. 사용하기 쉽고 사용하기에 도움이 되는 정보를 가지고 있는 메뉴를 제공해야 한다.

```
Welcome to Ewha Sky

============================================================
로그인 |
============================================================
아이디 입력 : myid
비번 입력 : password
로그인 성공

============================================================
홈 | 1. 마이페이지 2.예매하기  0. 종료
============================================================
입력 : 1

============================================================
마이페이지 | 1. 내 정보 2.예약 내역 3. 마일리지. 4. 쿠폰  5. 돌아가기 0. 종료
============================================================
입력 : 1
```

(14) 데이터베이스에 삽입(insert)을 하기 위한 인터페이스와 쿼리를 가지고 있어야 한다. 다음은 예약 테이블을 삽입하는 쿼리이다. 예약을 최종 예매하면 진행된다



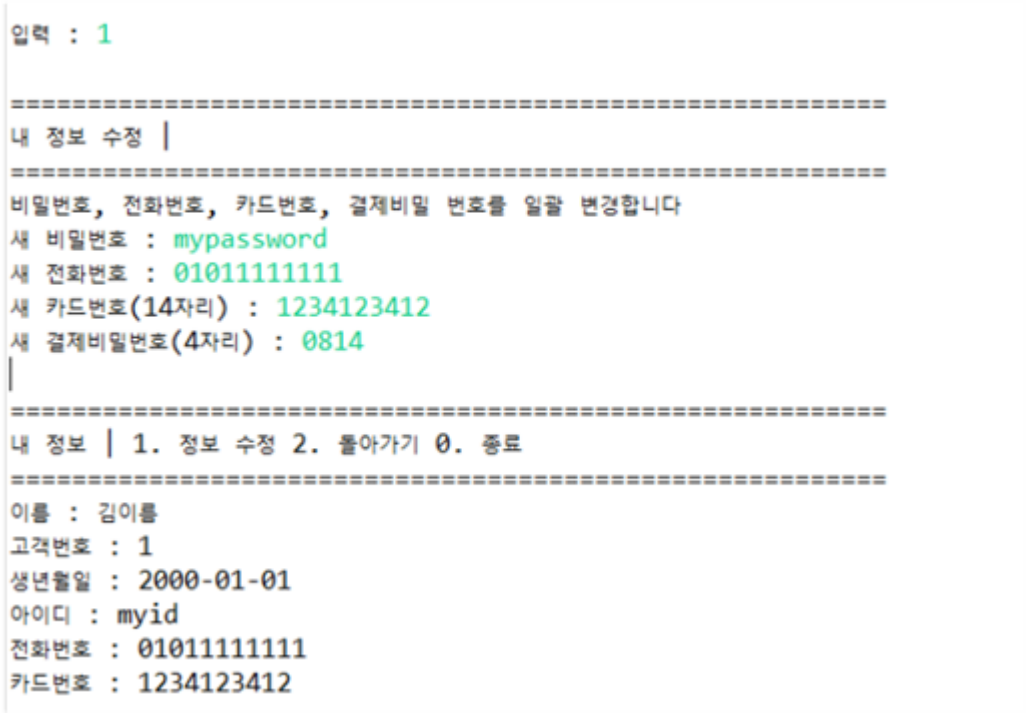
String insertReservation = "INSERT INTO DB2024_Reservations\r\n" + "(isCouponUsed, useMile,\r\n"

+ "earnMile, totalMile, payPrice, state,\r\n"

+ "DB2024_Customer_custid, DB2024_FlightName_flightNum, \r\n"

+ "DB2024_FlightName_goDate) VALUES (?,?,?,?,?,?,?,?,?)";

(15) 데이터베이스에 갱신(update)을 하기 위한 인터페이스와 쿼리를 가지고 있어야 한다. 다음은 고객 정보를 수정하는 쿼리이다.



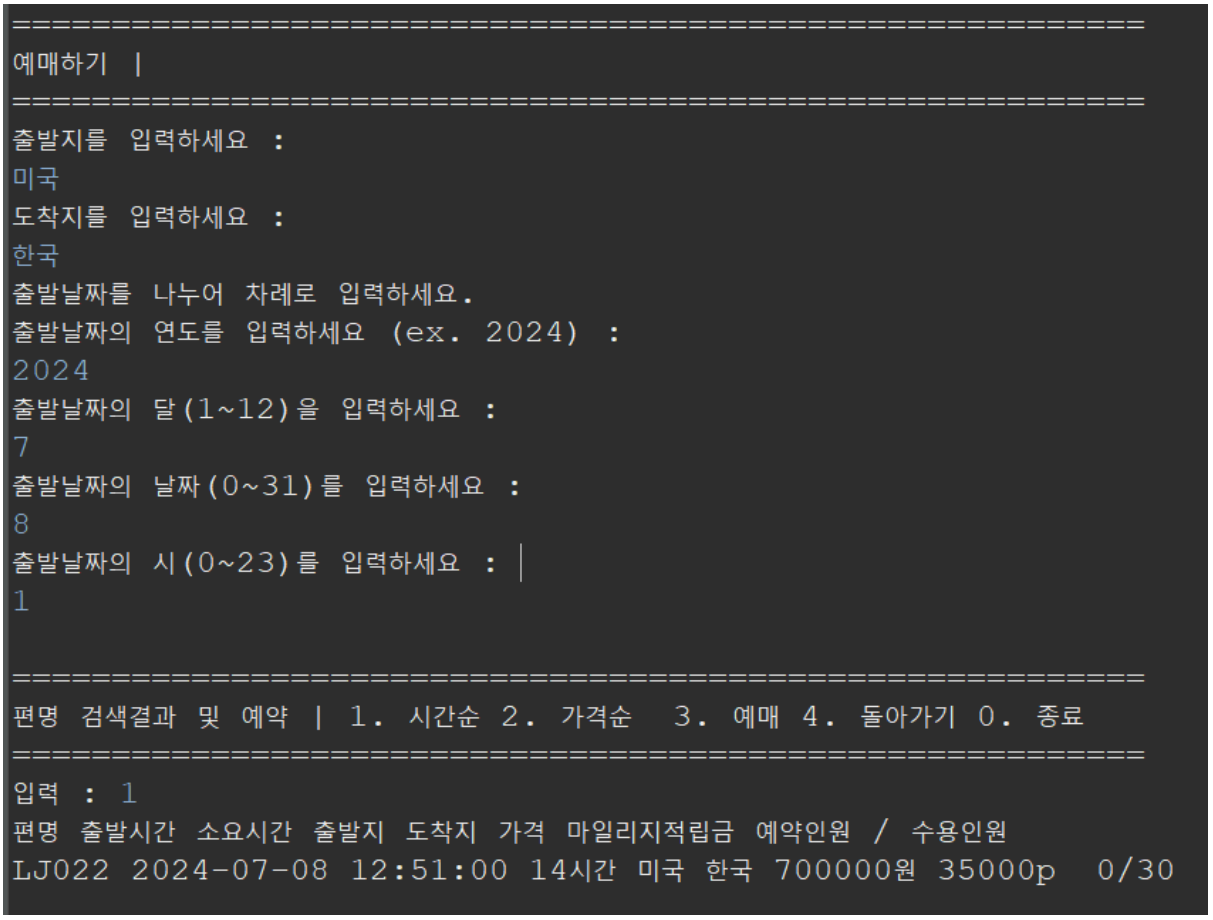

```
head("내 정보 수정 | ");
System.out.println("비밀번호, 전화번호, 카드번호, 결제비밀 번호를 일괄 변경합니다");
System.out.print("새 비밀번호 : ");
String pw = input.nextLine();
System.out.print("새 전화번호 : ");
String pn = input.nextLine();
System.out.print("새 카드번호(14자리) : ");
String cn = input.nextLine();
System.out.print("새 결제비밀번호(4자리) : ");
String cp = input.nextLine();
String query = "Update db2024_customer\n" + "Set PW='" + pw + "', phoneNum='" + pn + "',
cardNum='" + cn+ "', payPW='" + cp + "'\n" + "Where custid=" + custid + ";";
```

(16) 데이터베이스에 삭제(delete)를 하기 위한 인터페이스와 쿼리를 가지고 있어야 한다. 쿠폰 사용할 경우 쿠폰 테이블에서 삭제하는 쿼리이다.

```
예약을 취소할 예약 번호를 입력하세요.
19
일주일 내로 계좌로 1080000원이 환불 처리됩니다.
!!예약 취소로 인해 가장 오래된 쿠폰이 삭제되었습니다!!


============================================================
내 예약 | 1. 예약취소 2. 돌아가기 0. 종료
============================================================
예약번호 쿠폰사용여부 지불가격 현황 편번호 출발날짜 / 도착지 출발지
11 N 0원 결제 취소 KE240 2024-06-08 18:13:00 / 한국 미국

12 N 0원 결제 취소 KE052 2024-06-09 22:08:00 / 미국 한국

13 N 0원 결제 취소 KE005 2024-06-10 15:58:00 / 중국 한국

14 N 1400000원 결제 완료 KE250 2024-06-21 17:56:00 / 한국 미국

16 N 300000원 결제 완료 KE095 2024-06-22 11:46:00 / 중국 한국

17 N 170000원 결제 완료 KE480 2024-06-08 10:23:00 / 한국 일본

18 Y 220000원 결제 완료 KE055 2024-06-25 21:30:00 / 중국 한국

19 N 0원 결제 취소 OZ092 2024-06-08 10:20:00 / 미국 한국

20 N 0원 결제 취소 OZ075 2024-06-13 17:03:00 / 중국 한국

21 N 0원 결제 취소 OZ590 2024-07-08 11:47:00 / 한국 중국
```

```
if (isCouponUsed) {
isCouponUsedString = "Y";
String deleteCoupon = "DELETE FROM DB2024_Coupon WHERE couponID = '" +
couponID + "'";
stmt.executeUpdate(deleteCoupon);
}
```

(17) 데이터베이스에 검색(select)을 하기 위한 인터페이스와 쿼리를 가지고 있어야 한다. 다음은 사용자가 원하는 예약을 찾기 위해 검색하는 쿼리이다.




```
System.out.println("출발지를 입력하세요 : ");
String startCountry = s.next();
System.out.println("도착지를 입력하세요 : ");
String endCountry = s.next();
System.out.println("출발날짜를 나누어 차례로 입력하세요.");
System.out.println("출발날짜의 연도를 입력하세요 (ex. 2024) : ");
int year = s.nextInt();
System.out.println("출발날짜의 달(1~12)을 입력하세요 : ");
int month = s.nextInt();
System.out.println("출발날짜의 날짜(0~31)를 입력하세요 : ");
int day = s.nextInt();
System.out.println("출발날짜의 시(0~23)를 입력하세요 : ");
int hour = s.nextInt();
String query1 = "SELECT * FROM DB2024_vw_orderByTime \r\n"
                + "WHERE goDate >= '" + startDay + "' AND goDate <= '" + endDay + "' AND
startcountry = '" + startCountry + "' AND endcountry = '" + endCountry + "'" ;
```

(7) 요구조건 외의 팀만의 강점이 있다면 기술한다.

1. 자동 증가 기능 (AutoIncrement)

이화 스카이는 고객 관리 및 예약 시스템에서 자동 증가 기능을 활용하여 각 고객과 예약 건에 대해 고유한 식별자를 부여합니다. 이는 데이터베이스 관리 및 추적에 있어 혼동을 줄이고 효율성을 높입니다. 또한 중복되지 않는 고유 번호를 생성함으로써 데이터 무결성을 유지하고 오류를 방지할 수 있으며, 수작업을 줄여 운영 효율성을 극대화할 수 있습니다.

2. 예약 업데이트 시간

항공편 스케줄 및 수용인원에 대한 정보를 실시간으로 업데이트하여 고객이 최신 상태를 파악할 수 있도록 합니다. 이는 신뢰할 수 있는 최신 정보를 제공함으로써 고객 만족도를 높이고, 플랫폼에 대한 신뢰를 증대시킵니다.

3. 랜덤 할인금액

이화 스카이는 고객이 5번의 예약을 추가로 확정할 때마다 랜덤 할인 쿠폰을 발급합니다. 랜덤 할인 쿠폰 제공으로 고객의 예약 빈도를 증가시키고, 재이용률을 높입니다. 또한, 랜덤 요소를 도입하여 고객에게 즐거움과 기대감을 제공하고, 플랫폼 이용의 재미를 더합니다.

4. 마일리지

이화 스카이는 마일리지 적립 및 사용 현황을 한눈에 확인할 수 있는 기능을 제공합니다. 이를 통해 고객이 적립한 마일리지를 쉽게 확인하고 관리할 수 있습니다. 또한 적립된 마일리지를 효과적으로 사용하여 더 나은 여행 혜택을 누릴 수 있으며, 여러 항공사의 마일리지 정보를 통합하여 제공함으로써 사용자 편의성을 높입니다.

5. 예약 취소 시 쿠폰 회수

이화 스카이는 고객이 예약을 취소할 경우, 이미 발급된 랜덤 할인 쿠폰을 회수하는 시스템을 운영합니다. 예약 취소 시 회수 조건이 충족된 쿠폰을 회수함으로써 부정 사용을 방지하고, 공정한 혜택 제공을 유지합니다.

6. 샘플 편명 데이터 대량 생성

고객의 검색 기능 활용을 시뮬레이션 해보기 위해 많은 편명 데이터가 필요했기 때문에, random 함수를 활용한 파이썬 코드를 만들어 무한하게 많은 편명 튜플을 생성할 수 있도록 하였습니다. 그 결과, 2024년 6~7월에 해당하는 날짜들에 미국, 중국, 일본을 오가는 편명 데이터 100여개 이상을 한번에 삽입하여 데이터 베이스 구축을 하였습니다.

아래는 사용한 파이썬 코드 첨부합니다.

```
import random

"""
sample data
('LJ035', 'HL7101' , '2024-06-01 04:14:00', 0, '한국', '중국', 0, 0, 0),

flightNum CHAR(5),airplaneID VARCHAR(10), goDate,timeT,startCountry, endCountry ,
reservedNum,price,
mile


소요시간 timeT
한국 일본 2
한국 미국 14
한국 중국 3

가격 price
대한한공 소요시간*10
아시아나 소요시간*8
진에어 소요시간*5

마일리지 mile
대한한공 가격*0.1
아시아나 가격*0.07
진에어 가격*0.05

"""
airplaneId_list = ['HL7790', 'HL7387', 'HL7324', 'HL7100', 'HL7101','HL7102']
countries = ['일본', '중국', '미국']
timeT_list = [2, 3, 14]

reservedNum = 0
def flightNum_front(airplaneId):
    if airplaneId == 'HL7790':
        return 'KE'
```

```
    elif airplaneId == 'HL7387' or airplaneId ==  'HL7324':
        return 'OZ'
    else:
        return 'LJ'

def country_flightnum(country):
    if country == '한국':
        return 0
    elif country == '미국':
        return 2
    elif country == '일본':
        return 4
    elif country == '중국':
        return 5

def flight_price(time, airplaneId):
    if airplaneId == 'HL7790':
        return 10 * time * 10000
    elif airplaneId == 'HL7387' or airplaneId ==  'HL7324':
        return 8 * time * 10000
    else:
        return 5 * time * 10000

def flight_mileage(price , airplaneId):
    if airplaneId == 'HL7790':
        return (int)(price*0.1)
    elif airplaneId == 'HL7387' or airplaneId ==  'HL7324':
        return (int)(price * 0.07)
    else:
        return (int)(price * 0.05)



def add_parenthesis_comma(my_str):
    return "'" + my_str + "', "

def make_random_timestamp(number):
    date_list = []
    for i in range(number):

        YYYY = '2024'

        MM = str(random.randint(6, 8))
        if int(MM) < 10:
            MM = "0" + MM
```

```
        DD = str(random.randint(1, 30))
        if int(DD) < 10:
            DD = "0" + DD

        HH = str(random.randint(8, 22))
        if int(HH) < 10:
            HH = "0" + HH

        mm = str(random.randint(0, 59))
        if int(mm) < 10:
            mm = "0" + mm
        ss = '00'

        date_list.append( YYYY + "-" + MM +  "-" + DD  + " " + HH  + ":" + mm  + ":" + ss)
    date_list.sort()
    return date_list



#모든 항공기마다 30개씩 출력
for j in range(6):
    goDate_list = make_random_timestamp(30)
    for i in range(30):
        airplaneId = airplaneId_list[j]
        #한국이 도착인지 출발인지 정하기
        selection_list = [0, 1]
        korea_is_arrival = random.choice(selection_list)
        random.shuffle(selection_list)
        # 나라 선택
        select_country = random.randint(0, len(countries) - 1)
        othercountry = countries[select_country]
        timeT = timeT_list[select_country]
        if korea_is_arrival:
            startcountry = othercountry
            endcountry = '한국'

        else:
            endcountry = othercountry
            startcountry = '한국'
        flightNum = flightNum_front(airplaneId) + str(country_flightnum(endcountry)) + str(random.randint(0, 9)) +
str(country_flightnum(startcountry))
        price = flight_price(timeT, airplaneId)
```

```
mileage = flight_mileage(price, airplaneId)



final_flightnum = add_parenthesis_comma(flightNum)
final_airplaneId = add_parenthesis_comma(airplaneId)
final_goDate = add_parenthesis_comma(goDate_list[i])
final_timeT = str(timeT) + ", "
final_startcountry = add_parenthesis_comma(startcountry)
final_endcountry = add_parenthesis_comma(endcountry)
final_reservedNum = str(reservedNum) + ", "
final_price = str(price) + ", "
final_mileage =  str(mileage)


print( "(" + final_flightnum + final_airplaneId
    + final_goDate
    + final_timeT + final_startcountry
    + final_endcountry + final_reservedNum
    + final_price + final_mileage + "), ")
```

(8) 팀의 구성원의 담당 부분

팀원 이름	SQL	JAVA code	Report	발표	데모 비디오	기타
강민서	-쿼리 기초 틀 구성 -마일리지, 쿠폰 쿼리 작성 -고객,편명 데이터생성	-reserving 함수코드 작성 -reserved, mileage, coupon 함수코드 수정	-데이터베이스필요성설명 및 요구분석 -요구사항 중 뷰, 인덱스 정의함수작성 및 사용쿼리작성 -요구사항 중 갱신쿼리와 인터페이스부분 작성 -강점부분작성	-중간 발표 ppt 작성 - 기말 발표 ppt 작성	-시연영상 화면 녹화	
도연수	-쿼리 작성 (편명 검색과 정렬, 예약 및 쿠폰 삽입과 삭제) -편명 데이터 생성 코드 작성 -인덱스 수정 - 뷰 작성	-login 함수 초기 연결 -reserving, resultFlight, newCoupon,retrieveCoupon, select 구현 -cancelReserve 후반 수정 -트랜잭션 삽입 및 수정	X	-중간 발표 PPT 작성 -최종 발표 발표자	-시연 영상 흐름틀 초기 제작 -시연 영상 설명	-ERD 작성 - 전체 코드 병합 및 수정, sql 연동 확인

안채연	-쿼리 기초 틀 구상 -쿼리 작성 (로그인, 편명 검색과 정렬, 뷰, 예약 및 쿠폰 삽입과 삭제) -인덱스	-cancelReserve 구현 -트랜잭션 삽입 및 수정 -코드 테스트 및 수정	X	-중간 발표 발표자 -최종 발표 발표자	X	-프로그램 구체적인 흐름 구상(사용자 입력 조건별 페이지 전환)
양인경	-쿼리 기초 틀 구성 -고객, 편명 데이터 생성	-mileage 함수 -coupon 함수 -reserved 함수 내역 출력 -reserving 함수 검토 및 수정	-ER 다이어그램 및 설명 -스키마 다이어그램 작성 -요구사항 중 테이블 및 컬럼 그리고 투플의 개수에 대해 서술 -요구사항 중 기본 키, 외래 키, 제약 조건 등에 대해 서술 -요구사항 중 트랜젝션 작성 -요구 사항 중 조인 쿼리,	-기말 발표 ppt 5 -기말 발표 ppt 8~11	X	-ERD 작성

			중첩된 쿼리 부분 작성			
하지연	-예약 목록 뷰 생성 -고객, 편명 insert	-login 함수 -myInfo 함수 -chagneMyInfo 함수	- 자바 코드의 클래스(class)와 메서드(method)에 대한 설명 -(6)의 (12)~(17)	-중간 발표 발표자 -기말 발표 ppt 12~17 -기말 발표 ppt 최종 수정	X	- 자바 코드에서 전체적인 함수 틀 구상 -화면 구성 및 화면 이동 함수 구현