테이블 생성: memberTbl, ProductTbl

```
CREATE TABLE `shopdb`.`membertbl` (
  `memberID` CHAR(8) NOT NULL,
  `memberName` CHAR(5) NOT NULL,
  `memberAddress` CHAR(20) NULL,
  PRIMARY KEY (`memberID`));

CREATE TABLE `shopdb`.`producttbl` (
    `productName` CHAR(4) NOT NULL,
    `cost` INT NOT NULL,
    `makeDate` DATE NULL,
    `company` CHAR(5) NULL,
    `amount` INT NOT NULL,
PRIMARY KEY (`productName`));
```

값 넣기

```
INSERT INTO `shopdb`.`membertbl` (`memberID`, `memberName`, `memberAddress`) VALUES('Dang', '당탕이', '경기도 부천시 중동');
INSERT INTO `shopdb`.`membertbl` (`memberID`, `memberName`, `memberAddress`) VALUES('Jae', '지운이', '서울 은평구 증산동');
INSERT INTO `shopdb`.`membertbl` (`memberID`, `memberName`, `memberAddress`) VALUES('Han', '한주연', '인천 남구 주안동');
INSERT INTO `shopdb`.`membertbl` (`memberID`, `memberName`, `memberAddress`) VALUES('Sang', '상달이', '경기도 성남시 분당구');
INSERT INTO `shopdb`.`membertbl` (`memberID`, `memberName`, `memberAddress`) VALUES('Ezreal', '완상이', '경기도 김포시 걸포동');

INSERT INTO `shopdb`.`producttbl` (`productName`, `cost`, `makeDate`, `company`, `amount`) VALUES ('컴퓨터', '200', '2020-1-1', '삼성', '17');
INSERT INTO `shopdb`.`producttbl` (`productName`, `cost`, `makeDate`, `company`, `amount`) VALUES ('세탁기', '120', '2020-12-25', 'LG', '3');
INSERT INTO `shopdb`.`producttbl` (`productName`, `cost`, `makeDate`, `company`, `amount`) VALUES ('냉장고', '145', '2020-3-5', '대우', '22');
```

인덱스란
책 뒤에 찾아보기같은 개념, 빠르게 찾을 수 있음

index를 만들면 굉장히 빠르게 Data를 불러옴

```
-- 테이블 생성
CREATE TABLE shopdb.indexTBL (
    first_name VARCHAR(14),
    last_name VARCHAR(16),
    hire_date DATE
);

-- 테이블에 다른 D(employees)B의 다른 테이블(employees)의 성, 이름, 고용일 500개만 추가
INSERT INTO shopdb.indexTBL
  SELECT first_name, last_name, hire_date
  FROM employees.employees
  LIMIT 500;

SELECT * FROM indexTBL;
SELECT * FROM shopdb.indexTBL WHERE first_name = 'Mary';

-- INDEX 를 생성하면 빠르게 불러올 수 있다!
CREATE INDEX idx_indexTBL_firstname ON shopdb.indexTBL(first_name);
SELECT * FROM shopdb.indexTBL WHERE first_name = 'Mary';
```

뷰(View)
뷰 특징

가상 테이블, 진짜 테이블에 링크된 개념
뷰를 사용하면 데이터 보안과 안정성이 좋음(읽기 전용의 특징을 잘 살림)
뷰 생성

```
USE shopdb;
-- View 생성
CREATE VIEW uv_memberTBL
AS SELECT memberName, memberAddress FROM memberTBL;
-- View 조회
SELECT * FROM uv_memberTBL;
```

저장 프로시저(Stored Procedure)
개념: MySQL에서 제공해주는 프로그래밍 기능

```
-- 저장 프로시저 정의
DELIMITER //
CREATE PROCEDURE myProc()
BEGIN
    SELECT * FROM memberTBL WHERE memberName = '당탕이';
    SELECT * FROM productTBL WHERE productName = '냉장고';
END //

-- 실행
CALL myProc();
```

트리거(Trigger)

개념: 테이블에 부착되어 테이블에 특정 작업이 발생되면 실행되는 코드

예시: 회원Table의 회원 삭제 -> 트리거 발동 -> 삭제Table에 저장

```
-- 삭제 테이블 생성
CREATE TABLE deletedMemberTBL(
    memberID char(8),
    memberName char(5),
    memberAddress char(20),
    deleteDate date -- 삭제 날짜
);

-- TRIGGER 생성
DELIMITER //
CREATE TRIGGER trg_deletedMemberTBL  -- 트리거 이름
    AFTER DELETE  -- 삭제 후에 작동
    ON memberTBL  -- 트리거를 부착할 테이블
    FOR EACH ROW  -- 각 행마다 실행
BEGIN
    -- OLD 테이블의 내용을 백업테이블에 삽입
    INSERT INTO deletedMemberTBL
    VALUES (OLD.memberID, OLD.memberName, OLD.memberAddress, CURDATE());
END //

USE shopdb;
SELECT * FROM memberTBL;
DELETE FROM memberTBL WHERE memberName = '당탕이';
-- ERROR: 1175 Safe Update 가 뜬다면 안전하게(킷값으로만) 삭제 업데이트 설정이 켜져있어서임
-- Edit -> Preferences -> SQL Editor -> Safe Updates(가장 아래) 체크 해제
```