SELECT

샘플 데이터

- EMP 테이블 – 사원 테이블
  - ✓ EMPNO: 사원 번호 – 정수 4자리, 기본키
  - ✓ ENAME: 사원 이름 – 문자
  - ✓ JOB: 직무 – 문자
  - ✓ MGR: 관리자 사원 번호 – 정수 4자리
  - ✓ HIREDATE: 입사일 – DATE
  - ✓ SAL: 급여 – 실수 7자리이고 소수 2자리
  - ✓ COMM: 상여금 – 실수 7자리이고 소수 2자리(NULL 포함)
  - ✓ DEPTNO: 부서 번호 – 정수 2자리 이고 DEPT 테이블의 DEPTNO를 참조

샘플 데이터

- DEPT 테이블 – 부서 테이블
  - ✓ DEPTNO: 부서 번호 – 정수 2자리 이고 기본키
  - ✓ DNAME: 부서 이름 – 문자
  - ✓ LOC: 위치 – 문자

샘플 데이터

- SALGRADE – 호봉 테이블
  - ✓ GRADE: 호봉 – 숫자이고 기본키
  - ✓ LOSAL: 최저급여 – 숫자
  - ✓ HISAL: 최대급여 – 숫자

샘플 데이터

- TCITY – 도시 테이블
  - ✓ NAME: 도시이름 – 문자열이고 기본키
  - ✓ AREA: 면적 – 정수
  - ✓ POPU: 인구수 – 정수
  - ✓ METRO: 대도시 여부 – 문자
  - ✓ REGION: 도 – 문자

샘플 데이터

- TSTAFF – 직원 테이블
  - ✓ NAME: 이름 – 문자열이고 기본키
  - ✓ DEPART: 부서 이름 – 문자열
  - ✓ GENDER: 성별 – 문자열
  - ✓ JOINDATE: 입사일 – 날짜
  - ✓ GRADE: 직무 – 문자열
  - ✓ SALARY: 급여 – 정수
  - ✓ SCORE: 고과 점수 – 실수이고 소수 2자리

주석

❖ Maria DB의 주석

- ✓ 코드에 설명을 달거나 잠시 해당 부분의 실행을 막고 싶을 때 사용
- ✓ 한 줄 주석은 -- 이후부터 주석으로 처리
- ✓ -- 뒤에 바로 붙여서 쓰면 안되며 공백이 하나 이상 있어야 함

```
-- 한 줄 주석 연습
SELECT first_name, last_name, gender -- 이름과 성별 열을 가져옴
FROM employees;
```

- ✓ 여러 줄 주석은 /* */로 묶음

```
/* 블록 주석 연습
SELECT first_name, last_name, gender
FROM employees;
*/
```

데이터 조회

- SQL SELECT
  - ✓ SELECTION: 질의에 대해 RETURN하고자 하는 테이블의 행을 선택하기 위한 SQL의 연산
  - ✓ PROJECTION: 질의에 대해 RETURN하고자 하는 테이블의 열을 선택하기 위해 SQL의 연산
  - ✓ JOIN: 공유 테이블 양쪽의 열에 대해 링크를 생성하여 다른 테이블에 저장되어 있는 데이터를 함께 가져오기 위해 SQL의 join 기능을 사용할 수 있음

데이터 조회

❖ SELECT 기본 형식

```
SELECT      [DISTINCT] {*,column [Alias],...}
FROM        테이블명 ;
[WHERE      condition]
[GROUP BY   그룹핑 할 조건]
[HAVING     GROUP BY 이후의 condition]
[ORDER BY   {column, expression} [ASC | DESC]]
[LIMIT row_count [OFFSET offset]];
```

SELECT : 원하는 컬럼을 선택

FROM : 원하는 데이터가 저장된 테이블 명을 기술

WHERE : 조회되는 행을 제한(선택)

GROUP BY : 그룹화하고자 하는 컬럼 이름 나열

HAVING : 그룹화 한 이후의 조건

ORDER BY : 정렬을 위한 옵션(ASC:오름차순(Default),DESC내림차순)

LIMIT : 조회되는 데이터의 행 개수 제한 및 범위 제한

데이터 조회

- SELECT 기본 형식
  - ✓ 실행 순서 : FROM -> WHERE -> GROUP BY -> HAVING -> SELECT -> ORDER BY
  - ✓ 모든 열 선택
    - SELECT 키워드에 “*“ 을 사용하여 테이블의 열 데이터 모두를 조회
    - 오라클은 데이터를 입력한 순서대로 조회

데이터 조회

- SELECT 기본 형식
  - ✓ tCity 테이블의 전체 데이터 확인

```
SELECT * FROM tCity;
```

부산	765	342	y	경상
서울	605	974	y	경기
순천	910	27	n	전라
오산	42	21	n	경기
전주	205	65	n	전라
청주	940	83	n	충청
춘천	1116	27	n	강원
홍천	1819	7	n	강원

데이터 조회

❖ SELECT 기본 형식

✓ tStaff 테이블의 전체 데이터 확인

```
SELECT * FROM tStaff;
```

강감찬	영업부	남	2018-10-09	사원	320	56.00
김유신	총무부	남	2000-02-03	이사	420	88.80
논개	인사과	여	2010-09-16	대리	340	46.20
대조영	총무부	남	2020-07-07	차장	290	49.90
선덕여왕	인사과	여	2017-08-03	사원	315	45.10
성삼문	영업부	남	2014-06-08	대리	285	87.75
신사임당	영업부	여	2013-06-19	부장	400	92.00
안중근	인사과	남	2012-05-05	대리	256	76.50
안창호	영업부	남	2015-08-15	사원	370	74.20
유관순	영업부	여	2009-03-01	과장	380	
윤봉길	영업부	남	2015-08-15	과장	350	71.25
을지문덕	영업부	남	2019-06-29	사원	330	
이사부	총무부	남	2000-02-03	대리	375	50.00
이율곡	총무부	남	2016-03-08	과장	385	65.40
장보고	인사과	남	2005-04-01	부장	440	58.30
정몽주	총무부	남	2010-09-16	대리	370	89.50
정약용	총무부	남	2020-03-14	과장	380	69.80
허난설헌	인사과	여	2020-01-05	사원	285	44.50
홍길동	인사과	남	2019-08-08	차장	380	77.70
황진이	인사과	여	2012-05-05	사원	275	52.50

데이터 조회

- 특정 Column 선택
  - ✓ 테이블의 특정 Column을 검색하고자 할 경우 Column이름을 “,”로 구분하여 명시함으로써 특정 Column을 조회
  - ✓ 조회 순서는 SELECT문 뒤에 기술한 Column의 순서대로 조회

데이터 조회

❖ 특정 Column 선택

✓ tCity 테이블의 name, popu 필드 확인

```
SELECT name, popu
FROM tCity;
```

부산	342
서울	974
순천	27
오산	21
전주	65
청주	83
춘천	27
홍천	7

데이터 조회

❖ 특정 Column 선택

✓ tCity 테이블의 region, name, area 필드 확인

```
SELECT region, name, area
FROM tCity;
```

경상	부산	765
경기	서울	605
전라	순천	910
경기	오산	42
전라	전주	205
충청	청주	940
강원	춘천	1116
강원	홍천	1819

데이터 조회

- 특정 Column 선택
  - ✓ tStaff 테이블의 name, salary 필드 확인

```
SELECT name, salary
FROM tStaff;
```

강감찬	320
김유신	420
논개	340
대조영	290
선덕여왕	315
성삼문	285
신사임당	400
안중근	256
안창호	370
유관순	380
윤봉길	350
을지문덕	330
이사부	375
이율곡	385
장보고	440
정몽주	370
정약용	380
허난설헌	285
홍길동	380
황진이	275

데이터 조회

- ❖ 특정 Column 선택
  - ✓ 연습문제
    - ● tStaff 테이블에서 name, depart, grade 필드를 조회

데이터 조회

❖ 별명

- ✓ SELECT 명령이 출력하는 내용을 Result Set 또는 Row Set이라고 하는데 형태가 테이블 과 동일
- ✓ 원본 테이블의 일부만 읽어도 가로, 세로로 칸이 쳐진 도표 형태가 되기 때문에 SELECT가 만들어 내는 Result Set을 하나의 테이블처럼 사용
- ✓ Result Set의 필드 캡션은 테이블에서 정의한 이름과 동일한데 name 필드는 name이라고 출력하고 popu는 popu라고 출력하는데 테이블의 필드명은 구분 가능하고 입력하기 쉬운 짧은 명칭일 뿐이어서 사용자가 읽기에는 직관적이지 못한 경우가 있을 수 있는데 이럴 때는 필드에 대한 별명을 지정해서 Result Set의 헤더에 필드 이름 대신 별명을 출력
- ✓ 별명은 어디까지나 문자열일 뿐이므로 명칭 규칙에 영향을 받지 않는데 공백이나 기호는 물론이고 모든 문자를 자유롭게 표기할 수 있음
- ✓ 필드에 별명을 붙일 때는 다음 형식을 사용

  필드명 [AS] "별명"

- ✓ 필드명과 별명 사이에 전치사 AS를 넣는데 생략 가능
- ✓ 별명은 명칭이 아니며 공백이나 특수문자를 포함할 수 있어 큰 따옴표로 감싸되 평이한 단어라면 따옴표를 생략해도 상관없음
- ✓ 공백이나 특수 문자 또는 대문자가 있으면 이중 인용부호(“ ”)가 필요
- ✓ 별칭은 SELECT 이후에 수행되는 절에서만 사용할 수 있고 별칭 이므로 원래의 이름을 사용해도 되고 별칭을 사용해도 됨

데이터 조회

❖ 별명

✓ 별명 사용

```
SELECT name 도시명, area AS "면적(제곱Km)", popu AS "인구(만명)"
FROM tCity;
```

도시명	면적(제곱Km)	인구(만명)
부산	765	342
서울	605	974
순천	910	27
오산	42	21
전주	205	65
청주	940	83
춘천	1116	27
홍천	1819	7

데이터 조회

- 별명
  - ✓ 아래 문장은 에러인데 그 이유는 ?

```
SELECT name 도시, popu [인구(만명)]
FROM tCity;
```

데이터 조회

- 계산 값의 출력
  - ✓ 데이터가 조회되는 방식을 수정하거나 계산을 수행하고자 할 때 산술 표현식을 사용
  - ✓ 산술 표현식은 열 이름, 숫자 상수, 문자 상수, 산술 연산자를 포함할 수 있으며 연산자는 +(Add), -(Subtract), *(Multiply), /(Divide), DIV(몫), % 와 MOD(나머지) 사용 가능
  - ✓ SELECT 문장에서는 FROM 절을 제외한 절에서 사용할 수 있음
  - ✓ 산술 표현식이 하나 이상의 연산자를 포함한다면 일반적인 산술 연산자 우선 순위를 적용

데이터 조회

- 계산 값의 출력
  - ✓ 필드 목록에 계산식을 사용하면 테이블에 저장된 값을 가공하여 출력
  - ✓ popu 필드에 10000을 곱해서 출력

```
SELECT name, popu * 10000 AS "인구(명)"
FROM tCity;
```

name	인구(명)
부산	3420000
서울	9740000
순천	270000
오산	210000
전주	650000
청주	830000
춘천	270000
홍천	70000

데이터 조회

- 계산 값의 출력
  - ✓ 면적과 인구수를 이용해서 인구 밀도 조회

```
SELECT name, area, popu, popu * 10000 / area AS "인구밀도"
FROM tCity;
```

name	area	popu	인구밀도
부산	765	342	4470.5882
서울	605	974	16099.1736
순천	910	27	296.7033
오산	42	21	5000.0000
전주	205	65	3170.7317
청주	940	83	882.9787
춘천	1116	27	241.9355
홍천	1819	7	38.4827

데이터 조회

- ❖ 계산 값의 출력
  - ✓ 계산식 출력(Oracle에서는 FROM 절에 Dual을 추가해서 실행)

    SELECT 60 * 60 * 24;

    86,400
  - ✓ 연습문제
    - ● SELECT 문을 사용하여 1년은 몇 초인지 계산

데이터 조회

- ❖ 컬럼 연결
  - ✓ CONCAT
    - ● 서식: CONCAT(str1, str2, ...)
    - ● 설명
      - ◆ 인수를 연결한 결과로 나오는 스트링을 리턴
      - ◆ 한 개 또는 그 이상의 인수를 가질 수 있음

데이터 조회

❖ 컬럼 연결

✓ EMP 테이블에서 ENAME과 JOB을 묶어서 EMPLOYEES 로 조회

```
SELECT CONCAT(ENAME,' ', JOB) AS "EMPLOYEES"
FROM EMP;
```

```
SMITH|CLERK
ALLEN|SALESMAN
WARD|SALESMAN
JONES|MANAGER
MARTIN|SALESMAN
BLAKE|MANAGER
CLARK|MANAGER
SCOTT|ANALYST
KING|PRESIDENT
TURNER|SALESMAN
ADAMS|CLERK
JAMES|CLERK
FORD|ANALYST
MILLER|CLERK
```

데이터 조회

- DISTINCT
  - ✓ 특별히 명시되지 않았다면 SELECT 구문은 중복되지는 행을 제거하지 않고 Query 결과를 조회
  - ✓ 결과에서 중복되는 행을 제거하기 위해서는 SELECT 절에서 컬럼 이름 앞에 DISTINCT를 기술
  - ✓ DISTINCT 라는 키워드는 항상 SELECT 바로 다음에 기술
  - ✓ DISTINCT 뒤에 나타나는 컬럼 들은 모두 DISTINCT의 영향을 받음
  - ✓ DISTINCT 뒤에 여러 개의 컬럼을 기술하였을 때 나타나는 행은 컬럼의 조합들이 중복되지 않게 조회

데이터 조회

- DISTINCT
  - ✓ tCity 테이블에서 region 의 값을 중복을 제외하고 조회

```
SELECT region
FROM tCity;
```

```
경상
경기
전라
경기
전라
충청
강원
강원
```

데이터 조회

- DISTINCT
  - ✓ tCity 테이블에서 region 의 값을 중복을 제외하고 조회

```
SELECT DISTINCT region
FROM tCity;
```

경상
경기
전라
충청
강원

데이터 조회

- DISTINCT
  - ✓ tCity 테이블에서 region 의 값을 중복을 제외하고 조회

```
SELECT DISTINCT region
FROM tCity
ORDER BY region;
```

강원
경기
경상
전라
충청

데이터 조회

- DISTINCT
  - ✓ DISTINCT 뒤에 여러 개의 필드 나열

```
SELECT DISTINCT region, name
FROM tCity
ORDER BY region;
```

강원	춘천
강원	홍천
경기	서울
경기	오산
경상	부산
전라	순천
전라	전주
충청	청주

데이터 조회

- 정렬
  - ✓ SELECT 명령에 별 지정이 없을 경우 레코드의 출력 순서는 DBMS의 디폴트 순서를 따름
  - ✓ 오라클은 레코드의 입력 순서를 기억해 두고 그대로 가져오고 SQL Server 와 Maria DB는 기본키에 대해 오름차순으로 정렬
  - ✓ 관계형 DB에서 레코드의 물리적인 순서는 큰 의미가 없는 대신 출력할 때 ORDER BY 절로 정렬 순서를 원하는 대로 지정
  - ✓ 기본 형식
    - ORDER BY 필드 [ASC | DESC]
    - 오름차순의 경우는 ASC 그리고 내림차순의 경우는 DESC를 지정
    - 기본값이 ASC 이므로 ASC는 생략 가능
  - ✓ 디폴트 정렬은 오름차순
    - 숫자 값은 가장 적은 값이 먼저 조회(예 : 1 ~ 999)
    - 날짜 값은 가장 빠른 값이 먼저 조회(예 : 01-JAN-92 ~ 01-JAN-95)
    - 문자 값은 알파벳 순서로 조회(예 : A ~ Z ~ a ~ z)
    - NULL 값은 오름차순에서는 제일 나중에 그리고 내림차순에서는 제일 먼저

데이터 조회

- ❖ 정렬
  - ✓ ORDER BY
    - ● 인구 수에 따른 정렬

```
SELECT *
FROM tCity
ORDER BY popu;
```

홍천	1819	7	n	강원
오산	42	21	n	경기
순천	910	27	n	전라
춘천	1116	27	n	강원
전주	205	65	n	전라
청주	940	83	n	충청
부산	765	342	y	경상
서울	605	974	y	경기

데이터 조회

❖ 정렬

✓ ORDER BY

● 인구 수에 따른 정렬

```
SELECT *
FROM tCity
ORDER BY popu DESC;
```

서울	605	974	y	경기
부산	765	342	y	경상
청주	940	83	n	충청
전주	205	65	n	전라
순천	910	27	n	전라
춘천	1116	27	n	강원
오산	42	21	n	경기
홍천	1819	7	n	강원

데이터 조회

- 정렬
  - ✓ 2개 이상의 기준 필드를 지정할 수 있는데 이 경우에는 첫번째 기준 필드의 값이 같으면 두번째 기준 필드를 비교하여 정렬 순서를 결정
  - ✓ 필드 이름 대신에 필드의 순서 값을 설정하는 것도 가능
  - ✓ 정렬 기준 필드를 조회하지 않는 것도 가능 – 권장하지는 않음
  - ✓ 계산식을 이용한 정렬 가능

데이터 조회

❖ 정렬

✓ 지역 별로 정렬하고 같은 지역에 속한 도시 끼리는 이름의 내림차순으로 조회

```
SELECT region, name, area, popu
FROM tCity
ORDER BY region, name DESC;
```

강원	홍천	1819	7
강원	춘천	1116	27
경기	오산	42	21
경기	서울	605	974
경상	부산	765	342
전라	전주	205	65
전라	순천	910	27
충청	청주	940	83

데이터 조회

❖ 정렬

✓ 필드 순서를 이용한 정렬

```
SELECT *
FROM tCity
ORDER BY area;
```

오산	42	21	n	경기
전주	205	65	n	전라
서울	605	974	y	경기
부산	765	342	y	경상
순천	910	27	n	전라
청주	940	83	n	충청
춘천	1116	27	n	강원
홍천	1819	7	n	강원

데이터 조회

- 정렬
  - ✓ 필드 순서를 이용한 정렬

```
SELECT *
FROM tCity
ORDER BY 2;
```

오산	42	21	n	경기
전주	205	65	n	전라
서울	605	974	y	경기
부산	765	342	y	경상
순천	910	27	n	전라
청주	940	83	n	충청
춘천	1116	27	n	강원
홍천	1819	7	n	강원

데이터 조회

- 정렬
  - ✓ 정렬 기준 필드를 조회하지 않음

```
SELECT name
FROM tCity
ORDER BY popu;
```

홍천
오산
순천
춘천
전주
청주
부산
서울

데이터 조회

- 정렬
  - ✓ 계산 식을 이용한 정렬

```
SELECT name, popu * 10000 / area
FROM tCity
ORDER BY popu * 10000 / area;
```

홍천	38.4827
춘천	241.9355
순천	296.7033
청주	882.9787
전주	3170.7317
부산	4470.5882
오산	5000.0000
서울	16099.1736

데이터 조회

- 정렬
  - ✓ 연습문제: 직원 목록을 월급이 적은 사람부터 순서대로 출력하되 월급이 같다면 성취도가 높은 사람을 먼저 조회

데이터 조회

- WHERE 절
  - ✓ WHERE 절은 읽을 레코드의 조건을 지정
  - ✓ 필드 목록은 읽을 열을 지정하는데 비해 WHERE 절은 읽을 행을 지정
  - ✓ WHERE 절이 없으면 모든 레코드를 다 조회
  - ✓ SELECT 명령은 조건에 맞는 레코드를 검색하는 것이 주 기능이어서 대개의 경우 WHERE 절과 함께 사용
  - ✓ WHERE 절은 DELETE, UPDATE 등의 명령과 함께 삭제 및 변경할 레코드를 선택할 때도 사용
  - ✓ WHERE 절에는 레코드를 선택하는 조건을 기술하는데 필드와 특정 값을 비교하는 형식으로 작성

데이터 조회

- WHERE 절
  - ✓ 비교 연산자
    - A = B
    - A > B
    - A < B
    - A >= B
    - A <= B
    - A <> B, A != B, A ^= B
    - NOT Column_name = 같지 않다
    - NOT Column_name > 보다 크지 않다
    - <=> 2개의 값이 같으면 1 그렇지 않으면 0

데이터 조회

- WHERE 절
  - ✓ <> 와 !=는 같은 연산자
  - ✓ 숫자는 상수를 그냥 쓰지만 문자열과 날짜 상수는 작은 따옴표로 감싸야 함
  - ✓ 도시명이 서울이라는 조건은 아래처럼 작성하는데 MySQL 과 MariaDB는 큰 따옴표로 감싸도 에러는 아님

    ```
    SELECT * FROM tCity WHERE name = '서울';
    ```

  - ✓ SQL 키워드는 대소문자 구분없이 작성해도 되지만 필드 안에 저장된 값은 대소문자를 구분 (테이블 이름이나 필드 이름은 데이터베이스 종류마다 다르게 적용)
  - ✓ Maria DB 와 MySQL은 저장할 때는 대소문자를 구분하지만 비교할 때는 대소문자 구별을 하지 않음
  - ✓ metro 필드를 소문자 y와 비교해야 광역시인 서울, 부산을 출력

    ```
    SELECT * FROM tCity WHERE metro = 'y';
    ```

부산	765	342	y	경상
서울	605	974	y	경기

데이터 조회

- WHERE 절
  - ✓ tCity 테이블에서 area 가 1000보다 큰 데이터 조회

```
SELECT *
FROM tCity
WHERE area > 1000;
```

춘천	1116	27	n	강원
홍천	1819	7	n	강원

데이터 조회

- WHERE 절
  - ✓ tCity 테이블에서 area 가 1000보다 큰 데이터 의 name, area 조회

```
SELECT name, area
FROM tCity
WHERE area > 1000;
```

```
춘천    1116
홍천    1819
```

데이터 조회

- WHERE 절
  - ✓ 대소문자 구분 조회
    - 대소문자를 구분하도록 하려면 비교하려는 컬럼 이름 앞에 BINARY 키워드를 사용

```
SELECT * FROM tCity WHERE metro = 'Y';
부산  765    342    y    경상
서울  605    974    y    경기

SELECT * FROM tCity WHERE BINARY metro = 'Y';
```

데이터 조회

- WHERE 절
  - ✓ 대소문자 구분 조회
    - 콜레이션(Collation) 설정 변경
    - MariaDB와 MySQL의 기본 문자 집합(Character Set)인 utf8 또는 utf8mb4의 기본 콜레이션은 보통 **_ci (Case Insensitive, 대소문자 무시)**로 끝나는데 이 설정을 **_bin (Binary) 또는 _cs (Case Sensitive)**로 변경
    - 테이블 또는 컬럼 생성 시 테이블이나 특정 컬럼을 생성할 때 콜레이션을 utf8mb4_bin으로 지정하면 해당 컬럼에서 대소문자를 구분

```
CREATE TABLE 테이블이름 (
    컬럼이름 자료형 COLLATE utf8mb4_bin,
    ...
);
```

    - 기존 테이블의 콜레이션 변경

```
ALTER TABLE 테이블이름
MODIFY 컬럼이름 자료형 COLLATE utf8mb4_bin;
```

데이터 조회

- WHERE 절
  - ✓ 대소문자 구분 조회

```
ALTER TABLE tCity
MODIFY metro VARCHAR(50) COLLATE utf8mb4_bin;

SELECT * FROM tCity WHERE metro = 'Y';
```

데이터 조회

- ❖ WHERE 절
  - ✓ 연습문제
    - 인구가 10만명 미만인 도시의 이름 조회
    - 전라도에 있는 도시의 정보 조회
    - 월급이 400만원 이상인 직원의 이름 조회

데이터 조회

- WHERE 절
  - ✓ NULL 비교
    - NULL은 값이 입력되어 있지 않은 특수한 상태를 표현
    - 값을 알 수 없거나 아직 결정할 수 없다는 의미이며 0 과는 다름
    - NULL 여부를 표현할 때는 = 나 <>, != 대신에 is null 과 is not null 을 사용하는데 이유는 NULL은 NULL이라는 데이터를 저장하는 것이 아니고 NULL 상태 여부를 표시
    - tStaff 테이블에서 NULL 여부 표현 – 첫번째는 데이터가 조회되지 않음

```
SELECT * FROM tStaff WHERE score = NULL;
SELECT * FROM tStaff WHERE score IS NULL;
SELECT * FROM tStaff WHERE score IS NOT NULL;
```

데이터 조회

- WHERE 절
  - ✓ 논리 연산자
    - 두 개 이상의 조건을 동시에 점검할 때는 AND(&&), OR(||), XOR, NOT(!) 논리 연산자를 사용
    - AND는 두 조건이 모두 참 인 레코드를 검색하며 OR는 두 조건 중 하나라도 참 인 레코드를 검사
    - AND의 우선 순위가 OR 보다 높음
    - NOT 연산자는 표현식의 진위 여부를 반대로 변경
    - 인구 100만 이상 그리고 면적 700 이상인 도시를 조회

```
SELECT *
FROM tCity
WHERE popu >= 100 AND area >= 700;
```

부산 765 342 y 경상

데이터 조회

- ❖ WHERE 절
  - ✓ 논리 연산자
    - ● 우선 순위

```
SELECT *
FROM tCity
WHERE region = '경기' AND popu >= 50 OR area >= 500;
```

부산	765	342	y	경상
서울	605	974	y	경기
순천	910	27	n	전라
청주	940	83	n	충청
춘천	1116	27	n	강원
홍천	1819	7	n	강원

데이터 조회

- WHERE 절
  - ✓ 논리 연산자
    - ● 우선 순위

```
SELECT *
FROM tCity
WHERE popu >= 50 OR area >= 500 AND  region = '경기' ;
```

부산	765	342	y	경상
서울	605	974	y	경기
전주	205	65	n	전라
청주	940	83	n	충청

데이터 조회

- WHERE 절
  - ✓ 논리 연산자
    - NOT

```
SELECT *
FROM tCity
WHERE region != '경기';

SELECT *
FROM tCity
WHERE NOT(region = '경기');
```

부산	765	342	y	경상
순천	910	27	n	전라
전주	205	65	n	전라
청주	940	83	n	충청
춘천	1116	27	n	강원
홍천	1819	7	n	강원

데이터 조회

- ❖ WHERE 절
  - ✓ 논리 연산자
    - ● NOT

```
SELECT *
FROM tCity
WHERE region != '전라' AND metro != 'y';

SELECT *
FROM tCity
WHERE NOT(region = '전라' OR metro = 'y');
```

오산	42	21	n	경기
청주	940	83	n	충청
춘천	1116	27	n	강원
홍천	1819	7	n	강원

데이터 조회

- WHERE 절
  - ✓ 연습문제
    - 직원 목록에서 월급이 300 미만이면서 성취도는 60 이상인 직원이 누구인지 조회
    - 영업부의 여직원 이름을 조회

데이터 조회

- WHERE 절
  - ✓ LIKE
    - = 비교 연산자는 완전히 일치하는 조건식을 표현하는데 비해 LIKE 연산자는 패턴으로 부분 문자열을 검색
    - LIKE 문 의 패턴에는 와일드 카드 사용 가능
    - 와일드 카드

문자	설명
%	임의 개수의 임의 문자
_	하나의 임의 문자
[]	문자 리스트 중 하나의 문자와 대응
[^]	문자 리스트에 포함되지 않은 하나의 문자와 대응

    - 와일드 카드 문자를 검색하고자 할 때는 ESCAPE를 이용

데이터 조회

- WHERE 절
  - ✓ LIKE
    - ● 천 자가 들어가는 도시를 검색

```
SELECT *
FROM tCity
WHERE name LIKE '%천%';
```

순천	910	27	n	전라
춘천	1116	27	n	강원
홍천	1819	7	n	강원

데이터 조회

- WHERE 절
  - ✓ LIKE
    - ● 천 자가 들어가지 않는 도시를 검색

```
SELECT *
FROM tCity
WHERE name NOT LIKE '%천%';
```

부산	765	342	y	경상
서울	605	974	y	경기
오산	42	21	n	경기
전주	205	65	n	전라
청주	940	83	n	충청

데이터 조회

- WHERE 절
  - ✓ LIKE
    - 천 으로 끝나는 도시를 검색

```
SELECT *
FROM tCity
WHERE name LIKE '%천';
```

순천	910	27	n	전라
춘천	1116	27	n	강원
홍천	1819	7	n	강원

데이터 조회

- WHERE 절
  - ✓ LIKE
    - 천 으로 시작하는 도시를 검색

```
SELECT *
FROM tCity
WHERE name LIKE '천%';
```

데이터 조회

- ❖ WHERE 절
  - ✓ LIKE
    - ● tStaff 테이블에서 name 컬럼에 두번째 글자가 덕인 데이터 조회

```
SELECT *
FROM tStaff
WHERE name LIKE '_덕%';
```

선덕여왕 인사과 여 2017-08-03 사원 315 45.10

데이터 조회

- ❖ WHERE 절
  - ✓ LIKE
    - ● tStaff 테이블에서 name 컬럼의 글자 수가 4인 데이터의 name을 조회

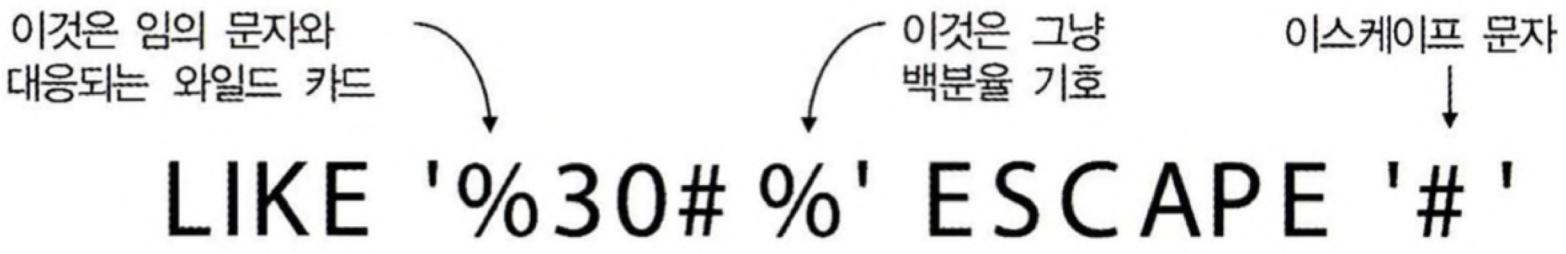
```
SELECT name
FROM tStaff
WHERE name LIKE '____';
```

선덕여왕
신사임당
을지문덕
허난설헌

데이터 조회

- ❖ WHERE 절
  - ✓ LIKE
    - ● sale에 30% 가 포함된 데이터 조회

      WHERE sale LIKE '%30#%' ESCAPE '#'



데이터 조회

- ❖ 조건문
  - ✓ 연습문제
    - ● 직원 목록에서 '정' 씨를 조회
    - ● 이름에 '신' 자가 포함된 직원을 조회

데이터 조회

- ❖ 조건문
  - ✓ BETWEEN
    - BETWEEN ~ AND 문은 BETWEEN 최소값 AND 최대값 형식으로 두 값 사이의 범위를 제한
    - 범위 조건은 수치 값에 대해 사용하지만 문자열이나 날짜 등에도 사용할 수 있음
    - 인구가 50 ~ 100 사이인 도시를 조회

```
SELECT *
FROM tCity
WHERE popu BETWEEN 50 AND 100;

SELECT *
FROM tCity
WHERE popu >= 50 AND popu <= 100;
```

데이터 조회

- ❖ 조건문
  - ✓ BETWEEN
    - ● 직원의 이름과 입사일로 범위 검색을 수행

```
SELECT *
FROM tStaff
WHERE name BETWEEN '가' AND '사';

SELECT *
FROM tStaff
WHERE joindate BETWEEN '20150101' AND '20180101';
```

데이터 조회

- ❖ 조건문
  - ✓ 연습문제
    - ● 면적 500~1000 사이의 도시 목록을 조회
    - ● 월급이 200 대인 직원의 목록을 조회

데이터 조회

- ❖ 조건문
  - ✓ IN
    - IN 연산자는 불연속적인 값 여러 개의 목록을 제공하여 이 목록과 일치하는 레코드를 검색
    - IN 연산자 뒤의 괄호 안에 콤마로 구분된 값 목록을 나열하여 이 중 하나에 해당하는지 점검하는데 값 개수에는 제한이 없어 얼마든지 많은 값을 넣을 수 있음
    - NOT 과 결합해서 사용 가능
    - region 필드가 경상 또는 전라 인 모든 도시를 조회

```
SELECT *
FROM tCity
WHERE region IN ('경상', '전라');

SELECT *
FROM tCity
WHERE region = '경상' OR region = '전라';
```

데이터 조회

- ❖ 조건문
  - ✓ IN
    - region 필드가 경상 또는 전라 가 아닌 모든 도시를 조회

```
SELECT *
FROM tCity
WHERE region NOT IN ('경상', '전라');
```

데이터 조회

- ❖ 조건문
  - ✓ IN
    - name 이 이 와 안으로 시작하는 단어를 조회

```
SELECT *
FROM tStaff
WHERE name LIKE '이%' OR name LIKE '안%';
```

데이터 조회

- ❖ 조건문
  - ✓ 연습문제
    - 총무부나 영업부에 근무하는 직원의 목록 조회
    - 인사과나 영업부에 근무하는 대리의 목록 조회
    - 차장급 이상의 여직원 목록을 조회

데이터 조회

- ❖ 정렬
  - ✓ ORDER BY
    - ● WHERE를 포함한 정렬

```
SELECT *
FROM tCity
WHERE region = '경기'
ORDER BY area;
```

데이터 조회

- ❖ 행의 개수 제한
  - ✓ LIMIT
    - SELECT 구문의 맨 뒤에 기재해서 행의 개수를 제한
    - ORACLE 에서는 ROWNUM 이라는 Pseudo Column 을 이용하고 MS-SQL에서는 TOP 을 이용
    - 기본 형식

      LIMIT [건너뛸 개수], 조회할 개수
      - ◆ 건너뛸 개수를 생략하면 0
      - ◆ LIMIT 조회할 개수 OFFSET 건너뛸 개수 형태로 입력하는 것도 가능

데이터 조회

- ❖ 행의 개수 제한
  - ✓ LIMIT
    - ● 면적이 넓은 상위 4개 도시 이름 조회

```
SELECT *
FROM tCity
ORDER BY area DESC
LIMIT 4;
```

데이터 조회

- ❖ 행의 개수 제한
  - ✓ LIMIT
    - ● 면적이 넓은 도시 중 앞의 2개는 건너뛰고 이후 3개의 데이터 조회

```
SELECT *
FROM tCity
ORDER BY area DESC
LIMIT 2, 3;

SELECT *
FROM tCity
ORDER BY area DESC
LIMIT 3
OFFSET 2 ;
```

데이터 조회

- ❖ 행의 개수 제한
  - ✓ 연습문제
    - ● 직원을 월급 순으로 내림차순 정렬한 후 12위에서 16위까지 조회

데이터 조회

- 검색 결과를 파일로 출력
  - ✓ 사용 방식
    - 결과를 외부 파일에 저장: SELECT ... INTO OUTFILE 'file_path';
    - 결과를 BLOB 데이터 유형과 같은 이진 데이터일 때 file_path 에서 지정하는 외부 파일로 저장: SELECT ... INTO DUMPFILE 'file_path';
    - 결과를 변수에 할당: SELECT ... INTO variable;
  - ✓ 형식
    - 기본 형식

      SELECT ... INTO OUTFILE 'file_name'

      [CHARACTER SET charset_name]

      [export_options]

      - file_path를 지정할 때 사용되는 디렉토리는 미리 존재해야 하는데 디렉토리가 존재하지 않으면 쿼리문을 실행할 때 오류 발생

데이터 조회

- ❖ 검색 결과를 파일로 출력
  - ✓ 형식
    - ● 옵션

```
export_options :
[{FIELDS I COLUMNS}
[TERMINATED BY ‘string’]
[[OPTIONALLY] ENCLOSED BY ‘char’]
[ESCAPED BY ‘char’]
]
[LINES
[STARTING BY ‘string’]
[TERMINATED BY ‘string’]
]
```

데이터 조회

❖ 검색 결과를 파일로 출력

✓ members 테이블의 모든 데이터를 파일로 내보내기

● 옵션

```
SELECT *
INTO OUTFILE
'/Users/adam/Documents/lecture/DataBase/MariaDB/result/members.dat'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
FROM members;
```

members.dat

```
"hong1","홍길동",\N,\N,\N,"인천 동구 송림동","2000-05-08","2","500","Y","2022-03-01 14:10:27",\N
"hong2","홍길동",\N,\N,\N,"서울 강남구 신사동","1990-01-05","9","1000","Y","2022-03-01 14:11:50",\N
"kim1","김갑수",\N,\N,\N,"인천 연수구 연수동","2003-07-01","1","0","Y","2022-03-01 14:12:39",\N
"park","박기자",\N,\N,\N,"경기 부천시","2002-09-30","3","0","Y","2022-03-01 14:13:16",\N
"seo","서갑돌",\N,\N,\N,"인천 동구","1998-03-10","1","0","Y","2022-03-01 14:08:41",\N
"Taeh","태현",\N,\N,\N,"경기 수원시","2002-10-15","4","0","Y","2022-03-01 14:15:10",\N
```

데이터 조회

- ❖ 검색 결과를 파일로 출력
  - ✓ 연습문제
    - goodsinfo 테이블을 외부 파일로 저장하기