3 / 12

- 데이터가 저장되어있는 테이블에서 필요한 데이터를 추출하기 위한 것.

- SQL 문제를 풀기 위해 기본적으로 알고 있어야 하는 문법
  - 기본검색 및 정렬
  - 그룹 제어
  - 분기문
  - 집합 연산

- SELECT [컬럼명들] (”,” 를 통해서 구분)
  - 추출하고 싶은 컬럼들을 보는 것.
- FROM [테이블명]
  - 추출하고 싶은 데이터가 있는 부분을 적는 것.
- WHERE [조건들] (and or 연산자를 통해서 구분)
  - FROM과 SELECT로 나열 된 데이터를 어떻게 추출할지 뽑는 것.
- ORDER BY [컬럼명들] (”,”를 통해서 구분)
  - 쉽표로 구분하여 나열이 가능하며, 오름차순 내림차순으로 한다.
- GROUP BY [컬럼명들] (”,”를 통해서 구분)
  - SELECT, FROM, WHERE를 통해 추출된 데이터를 특정 컬럼(그룹화)으로 만드는 것.
  - GROUP BY안에서 만 사용이 가능하지만, 필요에 따라 집계함수로 SELECT에 쓸 수 있다.
- HAVING [GROUP BY절에 해당하는 조건들] (”,”를 통해서 구분)
  - 그룹바이로 생성된 그룹에 대하여, 임의의 조건을 명시하는데 사용하는 조건
  - 그룹바이 절에 명시된 것만 사용가능하며, 집계함수도 가능

    - WHERE와 유사하지만, 집계함수를 사용한다는 점이 다르다.

- 분기문
- SIMPLE_CASE_EXPRESSION
    - (CASE [컬럼명] WHEN[비교값1] THEN [반환값1] … ELSE [모두 아니라면 반환값] END) AS [별칭 컬럼명]
- SEARCHED_CASE_EXPRESSION
    - (CASE WHEN [조건문1] THEN [반환값1] … ELSE [모두 아니라면 반환값] END) AS [별칭 컬럼명]

- 집합연산
- UNION
    - SELECT [컬럼1], [컬럼2], [컬럼3] FROM [테이블 명1] UNION SELECT [컬럼1], [컬럼2], [컬럼3] FROM [테이블 명2] → UNION은 중복 제외
- UNION ALL
    - SELECT [컬럼1], [컬럼2], [컬럼3] FROM [테이블 명1] UNION SELECT [컬럼1], [컬럼2], [컬럼3] FROM [테이블 명2] → UNION ALL은 중복 포함

- 순위집계
- RANK → 동일한 값에 대해서 동일한 등수를 매긴다.
    - SELECT RANK() OVER(PARTITION BY [그룹할 컴럼들] ORDER BY [순위를 매길때 사용할 컬럼들]) FROM [테이블명1]
    - Ex) 1, 2, 2, 2, 5, 6
- DENSE_RANK → 동일한 값에 대해 동일한 등수를 매기지만, 다음 순위는 순위를 건너 뛰지 않는다.
    - SELECT DENSE_RANK() OVER(PARTITION BY [그룹할 컬럼들] ORDER BY [순위를 매길 때 사용할 컬럼들]) FROM [테이블명1]
    - Ex) 1, 2, 2, 2, 3, 4
- ROW_NUMBER → 동일한 값에 대해서도 고유한 값을 매긴다.

    - SELECT ROW_NUMBER() OVER(PARTITION BY [그룹할 컬럼들] ORDER BY [순위를 매길 때 사용할 컬럼들]) FROM [테이블명1]
    - Ex) 1, 2, 3, 4, 5, 6

- 조인
- INNER JOIN → INNER를 제외하고, 사용해도 무방하다.
    - SELECT * FROM [테이블1] AS A INNER JOIN [테이블2] AS B ON A.KEY = B.KEY
    - 양쪽 모두 존재하는 데이터만 쓴다. 교집합
- LEFT OUTER JOIN
    - SELECT * FROM [테이블1] AS A LEFT OUTER JOIN [테이블2] AS B ON A.KEY = B.KEY
    - 왼쪽 테이블을 기준으로 오른쪽과 결합 할 때, 왼쪽에 없으면, NULL로 된다.
- RIGHT OUTER JOIN
    - SELECT * FROM [테이블1] AS A RIGHT OUTER JOIN [테이블2] AS B ON A.KEY = B.KEY
    - 왼쪽 테이블을 기준으로 오른쪽과 결합 할 때, 왼쪽에 없으면, NULL로 된다.
- FULL OUTER JOIN
    - SELECT * FROM [테이블1] AS A FULL OUTER JOIN [테이블2] AS B ON A.KEY = B.KEY
    - 좌측 테이블 1을 기준으로 테이블 2를 붙힌다. KEY가 없을 경우 NULL로 표출된다.
- SELF JOIN
    - SELECT * FROM [테이블1] AS A JOIN [테이블1] AS B ON A.NAME = B.MANAGER
    - 자기자신과 조인, 하나의 테이블에 같은 데이터가 존재하면서, 의미가 다르게 존재하는 경우 활용
    - 좌측 우측 테이블명이 동일하기 때문에 다르게 해주어야 오류가 나지 않는다.
- CROSS JOIN
    - SELECT * FROM [테이블1] AS A CROSS JOIN [테이블2] AS B

    - 두 테이블을 모두 결합하여, 모든 경우의 수가 나오게 된다.

- 집계 함수
- MAX ([컬럼명])
    - 명시된 컬럼 내 값들 중 최대값을 반환한다.
- MIN ([컬럼명])
    - 명시된 컬럼 내 값들 중 최소값을 반환한다.
- COUNT ([컬럼명])
    - 명시된 컬럼 내 값의 전체 행 수를 반환한다. (NULL값 제외)
- SUM ([컬럼명])
    - 명시된 컬럼의 데이터 타입이 숫자일 경우, 해당 컬럼 내 모든 데이터의 합을 반환한다. (NULL값은 제외)
- AVG ([컬럼명])
    - 명시된 컬럼의 데이터 타입이 숫자일 경우, 해당 컬럼 내 모든 데이터의 평균을 반환한다. (NULL값은 제외)

- 문자열 함수
- SUBSTRING(string, int, int)
    - 첫 번째 명시한 문자열의 부분 문자열을 잘라오기
- LTRIM(string), LTRIM(string, string) / RTRIM(string), RTRIM(string, string)
    - 명시한 문자열의 좌측/우측 공백을 제거한다. 특정 문자 제거
- LPAD(string, n, string) / RPAD(string, n, string)
    - 첫 번째 명시한 문자열에 길이가 n 이 되도록 좌측/우측 부터 세 번째 명시한 문자열로 채운 표현식을 반환한다.
- REPLACE(string, string_pattern, string_replacement)
    - 첫 번째 명시된 문자열 중 string_pattern에 해당하는 문자열을 string_replacement문자열로 변환한다.
- LENGTH(string)
    - 명시된 문자열의 길이를 구하여 반환한다.

- 날짜 함수
- NOW()
  - 현재의 날짜 및 시간을 출력
- AGE(timestamp, timestamp) / AGE(timestamp)
  - 두 날짜 사이의 시간차이를 계산 / 현재 날짜와 첫 번째 명시한 날짜의 시간 차이를 계산
- DATE_PART(text, timestamp)
  - 두 번째 명시한 timestamp에서 첫 번째 명시한 날짜 키워드 인자에 해당하는 값을 추출한다.
- DATE_TRUNC(text, timestamp)
  - 두 번째 명시된 timestamp에서 첫 번째 명시한 날짜 키워드 인자에 해당하는 값 이하의 날짜 데이터를 Default처리하고 반환한다.

- 그 외 함수
- TO_CHAR(timestamp, text)
  - 첫번째로 명시된 timestamp 값을 두번째 인자의 포맷 문자열로 변환하여 반환한다.
- COALESCE(value, ex1, ex2, ….)
  - 첫번째로 명시된 인자가 NULL일 경우 두번째 인자를 반환, 두번째 인자가 NULL일 경우 세번째 인자를 반환.. 순차적으로 반환 된다.
- CAST(source_type as target_type)
  - 첫번째 명시된 source_type을 두번째 인자로 명시된 target_type으로 변환하여 반환한다.
- ROUND(v numeric, s int)
  - 첫번째 명시된 v 값을 소수점 s자리까지 반올림 하고, s자리 미만은 버린다.