

1 REM Author :  
2 REM Date :  
3 REM Objective : Chapter 2. Basic **SELECT**  
4 REM Environment : Ubuntu Server 20.04 LTS, HeidiSQL 10.2.0, MySQL Community Server 5.7.34.0  
5

6 REM **SELECT**의 기능  
7 1. Selection : 조건검색, Row에 대한 필터링  
8 2. Projection : column 에 대한 필터링  
9 3. Join : 여러 테이블에서의 검색

10  
11  
12 REM **SELECT** Syntax

13  
14 **SELECT** [**DISTINCT** | **ALL**] { \* | column1, column2 [**AS** [alias]] | expr}  
15 **FROM** table\_name  
16 **WHERE** condition  
17 **ORDER BY** column [**ASC** | **DESC**];

18  
19 1. **SELECT** 절 다음에 질의하고 싶은 칼럼을 차례대로 나열한다. 이 때 여러 개의 칼럼 구분은 쉼표(,)로 한다.  
20 2. **FROM** 절 다음에는 조회할 테이블 이름을 적는다.  
21 3. \* : 모든 칼럼을 조회한다.  
22 4. **ALL** : 모든 결과 **ROW**를 보여준다.(기본값)  
23 5. **DISTINCT** : 중복된 **ROW**를 제외한 **ROW**를 보여준다.  
24 6. expr : **SQL** 함수를 사용하거나, 수학 연산을 포함한 표현식  
25 7. alias : 칼럼에 대한 별칭 사용.  
26 8. **Default Column Heading** : column 명이 대문자로 Display 된다.  
27 9. **Default Data Justification** : Number 값은 오른쪽 정렬, **Character** 와 **Date** 값은 왼쪽 정렬된다.

28  
29 REM 모든 열 선택  
30 **SELECT** \* **FROM** dept;

31  
32 **SELECT** \*  
33 **FROM** emp;

34  
35 **SELECT** \* **FROM** emp;

36  
37 ~~**SQL> SET** pagesize 1000 -- 1000줄을 한 페이지로 설정하는 **SQL\*Plus** 명령 실행~~  
38 ~~**SQL> /** -- **SQL\*Plus** buffer에 들어있는 **SQL** 문장을 다시 실행~~  
39 ~~**SQL> COL**[UMN] mgr **FOR**[MAT] 9999 -- 숫자 칼럼의 크기를 4자리로 조정하는 **SQL\*Plus** 명령 실행~~  
40 ~~**SQL> COL** ename **FORMAT** A8 -- 문자칼럼의 크기를 8자리로 조정하는 **SQL\*Plus** 명령 실행~~

41  
42  
43 REM 특정 열 선택  
44 1. 각 열의 구분은 " " 로 한다.

45 **SELECT** empno, ename, sal  
46 **FROM** emp;  
47  
48 **SELECT** empno, ename, job, mgr **FROM** emp;

49  
50  
51 REM 산술연산자 : 수학 연산 표현식

52 1. +, - : 음수, 혹은 양수를 나타내는 기호. 단항 연산자.  
53 2. \*(multiply), /(divide) : 곱하기, 나누기를 의미. 이항 연산자.  
54 3. +(add), -(subtract) : 더하기, 빼기를 의미. 이항 연산자.  
55 4. 연산자의 우선순위가 있다. 1 --> 2 --> 3  
56 5. 우선순위가 높은 연산 먼저 수행하며, 같은 우선순위의 연산자들을 왼쪽에서 오른쪽으로 순서대로 계산해 나간다.  
57 6. 괄호를 사용하여 우선순위를 조절할 수 있다.

58  
59 **SELECT** empno, ename, sal, sal + 100  
60 **FROM** emp;

61  
62 **SELECT** sal, -sal **FROM** emp;  
63 **SELECT** sal, sal \* 1.1 **FROM** emp;  
64 **SELECT** sal, comm, sal + comm **FROM** emp;  
65 **SELECT** sal, -sal + 100 \* -2 **FROM** emp;  
66 **SELECT** sal, (-sal + 100) \* -2 **FROM** emp;  
67 **SELECT** empno, ename, sal, sal \* 12 **FROM** emp;  
68 **SELECT** empno, ename, sal, sal \* 12 + comm **FROM** emp;  
69 **SELECT** empno, ename, sal, sal + comm \* 12 **FROM** emp;  
70 **SELECT** empno, ename, sal, (sal + comm) \* 12 **FROM** emp;

71  
72  
73 REM **NULL** value

74 1. **NULL** 이란?

```

75 1) 특정 행, 특정 열에 대한 아직 값을 알 수 없는 상태, 의미가 없는 상태를 표현
76 2) 이용할 수 없거나, 지정되지 않거나, 알 수 없거나, 적용할 수 없는 값
77 3) 아직 정의되지 않은 미지의 값
78 4) 현재 데이터를 입력하지 못하는 경우
79
80 2. NULL(ASCII 0)은 0(zero, ASCII 48) 또는 공백(blank, ASCII 32)과 다르다.
81
82 3. 연산의 대상에 포함되지 않는다.
83 4. NULL 값을 포함한 산술 연산 식의 결과는 언제나 NULL 이다.
84 5. NOT NULL 또는 Primary Key 제약조건이 걸린 칼럼에서는 NULL VALUE가 나타날 수 없다.
85 6. NULL 인 칼럼은 Length 가 0 이므로 데이터를 위한 물리적 공간을 차지 하지 않는다.
86
87 SELECT empno, job, comm FROM emp;
88 --NULL 인 값은 비어있는 것으로 표현된다.
89 --job이 salesman인 직원들에게만 커미션이 적용되며, 사번 7844인 직원의 커미션은 0이다.
90
91 SELECT empno, ename, sal * 12 + comm
92 FROM emp;
93 --comm 값이 NULL 인 경우 연봉은 얼마인가? 연봉 계산한 수식의 column heading은 어떻게 나타나는가?
94 --comm 값이 NULL 인 row 의 경우 (sal + comm) * 12를 하면 결과도 모두 NULL 이 된다.
95 --또한, expression 전체가 column heading 으로 나타난다.
96
97
98 REM IFNULL function
99 1. NULL 값을 어떤 특정한 값으로 치환할 때 사용
100 2. 치환할 수 있는 값의 형태는 숫자형, 문자형, 날짜형 모두 가능
101
102 3. Syntax
103 IFNULL(expr1, expr2)
104 --If expr1 is not NULL, IFNULL() returns expr1; otherwise it returns expr2.
105 --expr1 : NULL
106 --expr2 : 치환값
107 --expr1값이 NULL 아니면 expr1 값을 그대로 사용
108 --만약 expr1 값이 NULL이면, expr2 값으로 대체
109
110 4. 예
111 SELECT IFNULL(1, 0); --> 1
112 SELECT IFNULL(NULL, 10); --> 10
113 SELECT IFNULL(1/0, 10); --> 10
114 SELECT IFNULL(1/0, 'yes') ---> yes
115
116 IFNULL(comm, 0)
117 IFNULL(hiredate, '12/09/04')
118 IFNULL(job, 'No Job')
119
120 --위에서 연봉을 구하는 Query 를 NVL 함수를 사용하여 제대로 나올 수 있도록 고쳐보자.
121 SELECT empno, ename, sal, comm, sal * 12 + IFNULL(comm, 0)
122 FROM emp;
123
124 SELECT empno, comm, IFNULL(comm, 0)
125 FROM emp;
126
127 SELECT IFNULL(mgr, 'No Manager')
128 FROM emp;
129
130
131 REM Alias 별칭
132 1. column header 에 별칭을 부여 할 수 있다.
133 2. SELECT 절에 expression 을 사용할 때 도움이 된다.
134 3. 열 이름 바로 뒤에 기술한다. 또는 열이름과 별칭 사이에 AS를 사용할 수 있다.
135 4. 별칭에 공백이나 특수문자나 한글사용할 때, 대소문자를 기술할 때(기본값은 모두 대문자)에는 "" 로 기술한다.
136
137 SELECT empno 사번 FROM emp;
138 SELECT sal * 12 연봉 FROM emp;
139 SELECT sal * 12 annual_salary FROM emp;
140 SELECT sal * 12 Annual_Salary FROM emp;
141 SELECT sal * 12 Annual Salary FROM emp; --Error 발생
142 SELECT ename "Name" , sal AS "Salary", sal * 12 + IFNULL(comm, 0) AS "Annual Salary"
143 FROM emp;
144
145
146 REM Concatenation Operator (연결 연산자)
147 1. Oracle에서는 문자열 리터럴을 이을 때에는 '||' 를 사용한다.
148 2. 하지만 MySQL에서는 연결연산자( || )가 없기 때문에 CONCAT()를 사용한다.

```

149 3. **character** string 들을 연결하여 하나의 결과 string 을 만들어 낸다.

150

```
151 SELECT CONCAT(empno, ' ', ename) FROM emp;  
152 SELECT CONCAT(empno, ' ', ename, ' ', hiredate) FROM emp;  
153 --number 이나 date값은 default 형태의 character 값으로 자동 변환한 후 연결된다.
```

154

155

156 REM Literals (상수)

157 1. Literal 은 상수 값을 의미.

158 2. **Character** literal 은 작은 따옴표로 묶고, **Number** literal 은 따옴표 없이 표현한다.

159 3. **Character** literal을 작은 따옴표로 묶어 주어야 MySQL Server 는 keyword나 객체 이름을 구별할 수 있다.

160

```
161 SELECT CONCAT('Emp# of ', ename, ' is ', empno) FROM emp;  
162 SELECT CONCAT(dname, ' is located at ', loc) FROM dept;  
163 SELECT CONCAT(ename, ' is a ', job) AS "Employee" FROM emp;  
164 SELECT CONCAT(ename, ' ', sal) FROM emp;  
165 SELECT CONCAT(ename, ' is working as a ', job) FROM emp;  
166 SELECT 'Java is a language.' FROM emp; --14번 출력  
167 SELECT 'Java is a language.' FROM dept; --4번 출력  
168 SELECT 'Java is a language.';
```

169

170

171 REM Duplicate **Values**(중복 행 제거하기)

172 1. 일반 Query는 **ALL** 을 사용하기 때문에 중복된 행이 출력된다.

173 2. **DISTINCT** 를 사용하면 중복된 행의 값을 제거한다.

174 3. **DISTINCT** 는 **SELECT** 바로 뒤에 기술한다.

175 4. **DISTINCT** 다음에 나타나는 **column**은 모두 **DISTINCT** 에 영향을 받는다.

176

```
177 SELECT job FROM emp;  
178 SELECT ALL job FROM emp;  
179 SELECT DISTINCT job FROM emp;  
180 SELECT deptno FROM emp;  
181 SELECT DISTINCT deptno FROM emp;  
182 SELECT deptno, job FROM emp;  
183 SELECT DISTINCT deptno, job FROM emp;
```

184

185

186 REM **WHERE** 절

187 1. 사용자들이 자신이 원하는 자료만을 검색하기 위해서

188 2. Syntax

189

```
190 SELECT column...  
191 FROM table_name  
192 WHERE conditions;
```

193

194 3. **WHERE** 절을 사용하지 않으면 **FROM** 절에 명시된 **table**의 모든 **ROW**를 조회하게 된다.

195 4. **table**내의 특정 **row**만 선택하고 싶을 때 **WHERE** 절에 조건식을 사용한다.

196 5. MySQL Server 는 **table**의 **row**를 하나씩 읽어 **WHERE** 절의 조건식을 평가하여 **TRUE**로 만족하는 것만을 선택한다.

197 6. condition을 평가한 결과는 **TRUE, FALSE, NULL** 중의 하나이다.

198 7. condition 내에서 **character** 와 **date** 값의 literal은 작은 따옴표를 사용하고, **NUMBER** 값은 그대로 사용한다.

199 8. condition 에서 사용하는 **character** 값은 대소문자를 구별하지 않는다.

200 1) **WHERE** ename = 'JAMES';

201 2) **WHERE** ename = 'james';

202

203 9. **date** 타입의 변경은 **DATE\_FORMAT()**를 사용한다.

204 10. **WHERE** 는 **FROM** 다음에 와야 한다.

205 11. **WHERE** 절에 조건이 없는 FTS(**Full Table Scan**) 문장은 **SQL** Tunning의 1차적인 검토 대상이 된다.

206 12. **WHERE** 조건절의 조건식은 아래 내용으로 구성된다.

207 -**Column** 명(보통 조건식의 좌측에 위치)

208 -비교 연산자

209 -문자, 숫자, 표현식(보통 조건식의 우측에 위치)

210 -비교 **Column**명 (**JOIN** 사용시)

211

212

213 REM 비교연산자

214 --<, >, <=, >=, =, !=, <>(같지 않다)

215

216 --직위가 **CLERK** 인 사원의 이름과 직위 및 부서번호를 출력하시오.

```
217 SELECT ename, job, deptno  
218 FROM emp  
219 WHERE job = 'CLERK';
```

220

```
221 SELECT empno, ename, job  
222 FROM emp
```

```

223 WHERE empno = 7934;
224
225 SELECT empno, ename, job, hiredate
226 FROM emp
227 WHERE hiredate = '1981-12-03';
228
229 SELECT empno, ename
230 FROM emp
231 WHERE ename = 'JAMES';
232
233 SELECT empno, ename
234 FROM emp
235 WHERE ename = 'james';
236
237 SELECT dname
238 FROM dept
239 WHERE deptno = 30;
240
241 SELECT ename, sal
242 FROM emp
243 WHERE sal >= 1500;
244
245 --1983년 이후에 입사한 사원의 사번, 이름, 입사일을 출력하시오.
246 SELECT empno, ename, hiredate
247 FROM emp
248 WHERE hiredate >= '1983-01-01';
249
250 --급여가 보너스(comm) 이하인 사원의 이름, 급여 및 보너스를 출력하시오
251 SELECT ename, sal, comm
252 FROM emp
253 WHERE sal <= IFNULL(comm, 0);
254
255 --10번 부서의 모든 사람들에게 급여의 13%를 보너스로 지급하기로 했다. 이름, 급여, 보너스 금액, 부서번호를 출력하시오.
256 SELECT ename, sal, sal * 0.13, deptno
257 FROM emp
258 WHERE deptno = 10;
259
260 --30번 부서의 연봉을 계산하여, 이름, 부서번호, 급여, 연봉을 출력하라. 단, 연말에 급여의 150%를 보너스로 지급한다.
261 SELECT ename, deptno, sal, sal * 12 + IFNULL(comm, 0) + sal * 1.5 AS "년봉"
262 FROM emp
263 WHERE deptno = 30;
264
265 --부서번호가 20인 부서의 시간당 임금을 계산하시오. 단, 1달의 근무일수는 12일이고, 1일 근무시간은 5시간이다. 출력양식은 이름, 급여,
시간당 임금을 출력하라.
266 SELECT ename, sal, sal / 12 / 5
267 FROM emp
268 WHERE deptno = 20;
269
270 --모든 사원의 실수령액을 계산하여 출력하시오. 단, 이름, 급여, 실수령액을 출력하시오. (실수령액은 급여에 대해 10%의 세금을 뺀 금액)
271 SELECT ename, sal, sal - sal * 0.1 AS "실수령액"
272 FROM emp;
273
274 --사번이 7788인 사원의 이름과 급여를 출력하시오.
275 --급여가 3000이 넘는 직종을 선택하시오.
276 --PRESIDENT를 제외한 사원들의 이름과 직종을 출력하시오.
277 --BOSTON 지역에 있는 부서이 번호와 이름을 출력하시오.
278
279
280 REM 논리연산자
281 --AND(&&), OR(||), NOT(!)
282
283 --사원테이블에서 급여가 1000불이상이고, 부서번호가 30번인 사원의 사원번호, 성명, 담당업무, 급여, 부서번호를 출력하시오.
284 SELECT empno, ename, job, sal, deptno
285 FROM emp
286 WHERE sal >= 1000 AND deptno = 30;
287
288 --사원테이블에서 급여가 2000불이상이거나 담당업무가 매니저인 사원의 정보중 사원번호, 이름, 급여, 업무를 출력하시오.
289 SELECT empno, ename, sal, job
290 FROM emp
291 WHERE sal >= 2000 OR job = 'MANAGER';
292
293
294 REM SQL 연산자
295 1. BETWEEN A AND B : A보다 같거나 크고, B보다 작거나 같은

```

296 2. **IN(list)** : LIST 안에 있는 멤버들과 같은  
297 3. A **LIKE** B [**ESCAPE** 'C']: A가 B의 패턴과 일치하면 **TRUE**, 보통 %, \_ 연산자와 같이 사용, **escape** 을 사용하면 B의 패턴 중에서 C  
를 상수로 취급한다.  
298 4. **IS NULL / IS NOT NULL : NULL** 여부를 테스트

299  
300 1)**BETWEEN A AND B**  
301 --사원테이블에서 월급이 1300불에서 1500불까지의 사원정보중 성명, 담당업무, 월급을 출력하시오.  
302 **SELECT** ename, job, sal  
303 **FROM** emp  
304 -- WHERE sal >= 1300 AND sal <= 1500;  
305 **WHERE** sal **BETWEEN** 1300 **AND** 1500;  
306  
307 **SELECT** ename, job, sal  
308 **FROM** emp  
309 **WHERE** sal **BETWEEN** 1500 **AND** 1300;  
310 --반드시 작은 값이 먼저 나와야 한다.  
311  
312 **SELECT** ename **FROM** emp  
313 **WHERE** hiredate **BETWEEN** '1982-01-01' **AND** '1982-12-31';  
314  
315 --급여가 2000 에서 3000 사이인 사원을 출력하시오.  
316 **SELECT** ename, job, sal **FROM** emp  
317 **WHERE** sal **BETWEEN** 2000 **AND** 3000;  
318  
319  
320 2) **IN**  
321 --사원테이블에서 업무가 회사원, 매니저, 분석가인 사원의 이름, 업무를 출력하시오.  
322 **SELECT** ename **AS** "이름", job  
323 **FROM** emp  
324 -- WHERE job = 'CLERK' OR job = 'MANAGER' OR job = 'ANALYST';  
325 **WHERE** job **IN**('CLERK', 'MANAGER', 'ANALYST');  
326  
327 --관리자의 사원번호가 7902, 7566, 7788인 모든 사원의 사원번호, 이름, 급여 및 관리자의 사원번호를 출력하시오.  
328  
329 **SELECT** dname **FROM** dept **WHERE** deptno **IN**(10,20);  
330  
331 **SELECT** ename **FROM** emp  
332 **WHERE** job **IN**('ANALYST', 'CLERK')  
333  
334 --BOSTON 이나 DALLAS 에 위치한 부서를 출력하시오.  
335 **SELECT** dname, loc **FROM** dept  
336 **WHERE** loc **IN**('BOSTON', 'DALLAS');  
337  
338 --30, 40번 부서에 속하지 않는 사원들을 출력하시오.  
339 **SELECT** ename, deptno **FROM** emp  
340 **WHERE** deptno **NOT IN**(30,40);  
341  
342 --DALLAS 의 20번 부서, 또는 CHICAGO의 30번 부서를 출력하시오.  
343 **SELECT** \* **FROM** dept  
344 **WHERE** (deptno, loc) **IN** (20, 'DALLAS'), (30, 'CHICAGO'));  
345  
346  
347 3)**LIKE(%, \_)**  
348 --Wildcard : %(0개 이상의 문자 대표), \_ (1개의 문자 대표)  
349 --Wildcard 문자를 일반 문자로 사용하고 싶을 때 **ESCAPE** 을 사용한다.  
350 --**ESCAPE** 문자 바로 뒤에 사용된 Wildcard 문자는 일반 문자로 취급한다.  
351  
352 **SELECT** ename, job, hiredate **FROM** emp  
353 -- WHERE hiredate LIKE '1987%';  
354 **WHERE** hiredate >= '1987-01-01';  
355  
356 **SELECT** dname **FROM** dept  
357 **WHERE** dname **LIKE** 'A%';  
358  
359 --이름이 A로 시작하는 사원을 출력하시오.  
360 **SELECT** ename **FROM** emp  
361 **WHERE** ename **LIKE** 'A%';  
362  
363 --사번이 8번으로 끝나는 사원을 출력하시오.  
364 **SELECT** empno, ename **FROM** emp  
365 **WHERE** empno **LIKE** '%8';  
366  
367 --1982에 입사한 사원을 출력하시오.  
368 **SELECT** ename, hiredate **FROM** emp

```

369 -- WHERE hiredate LIKE '1982%';
370 -- WHERE hiredate BETWEEN '1982-01-01' AND '1982-12-31';
371 WHERE hiredate >= '1982-01-01' AND hiredate <= '1982-12-31';
372
373 SELECT empno, ename
374 FROM emp
375 WHERE ename LIKE 'MILLE_';
376
377 SELECT empno, ename
378 FROM emp
379 WHERE ename LIKE '%$_TEST' ESCAPE '$';
380
381
382 4) IS NULL / IS NOT NULL
383 --column 의 NULL 여부를 판단할 때에는 반드시 'IS NULL' 혹은 'IS NOT NULL' 연산자를 사용한다.
384 SELECT ename FROM emp
385 WHERE comm IS NULL;
386 WHERE comm IS NOT NULL;
387
388 --comm 지급 대상인 사원을 출력하시오.
389 SELECT ename, comm FROM emp
390 WHERE comm IS NOT NULL;
391
392 SELECT ename, mgr
393 FROM emp
394 WHERE mgr IS NULL;
395
396
397 REM 연산자 우선순위
398 1. !
399 2. -, ~, 괄호 ( )
400 3. ^
401 4. *, /, DIV, %, MOD : 산술연산자
402 5. -, + : 산술연산자, ||
403 6. <<, >>
404 7. &
405 8. |
406 9. =, !=, >=, >, <=, <, <>, !=, IS, LIKE, IN
407 10. BETWEEN, CASE, WHEN, THEN, ELSE
408 11. NOT
409 12. AND, &&
410 13. XOR
411 14. OR, ||
412
413 SELECT ename, job FROM emp
414 WHERE NOT job = 'ANALYST';
415
416 SELECT ename, sal, deptno FROM emp
417 WHERE sal > 2500 AND deptno = 20;
418
419 SELECT deptno, dname FROM dept
420 WHERE deptno = 10 OR deptno = 20;
421
422 --업무가 SALESMAN 이거나 업무가 MANAGER 이고, 급여가 1300불이상인 사람의 직원번호, 이름, 업무, 급여를 출력하시오.
423 SELECT empno, ename, job, sal
424 FROM emp
425 WHERE (job LIKE 'S%' OR job LIKE 'M%') AND sal >= 1300;
426
427 --직종이 CLERK 인 직원 중에서 급여가 1000 이상인 직원을 출력하시오.
428 SELECT ename, job, sal FROM emp
429 WHERE job = 'CLERK' AND sal >= 1000;
430
431
432 REM ORDER BY
433 1. 기본적으로 데이터는 정렬되지 않는다.
434 2. 같은 쿼리를 수행할 때마다 결과가 다르게 나올 수 있다.
435 3. 별칭은 정렬에 영향을 주지 않는다.
436 4. Syntax
437
438 SELECT column_list
439 FROM table
440 [WHERE conditions]
441 [ORDER BY column[, column] {ASC | DESC}];
442

```

```
443 5. 특징
444 1) 기본적으로 오름차순정렬한다.
445 --숫자인경우 ( 1 --> 999)
446 --날짜인경우 (옛날 --> 최근)
447 --문자인경우 (알파벳순서, 유니코드순)
448 2) NULL 은 오름차순일 경우는 제일 처음에, 내림차순인 경우에는 제일 마지막에 출력
449
450 6. ORDER BY 절에 정렬의 기준이 되는 column 을 여러개 지정할 수 있다. 첫번째 column 으로 정렬한 다음, 그 column 값이 같은
451 row 들에 대해서는 두 번째 column 값으로 정렬한다.
452 7. 오름차순(ASC) 정렬이 기본이며, 내림차순으로 정렬하고자 할 때에는 DESC를 사용한다.
453 8. ORDER BY 절에 column 이름 대신 positional notation 을 사용할 수도 있다. Position number 는 SELECT 절에서의 column
454 순서를 의미한다.
455
456 --입사일자 순으로 정렬하여 사원번호, 이름, 입사일자를 출력하시오.
457 SELECT empno, ename, hiredate
458 FROM emp
459 ORDER BY hiredate DESC;
460
461 --부서번호가 20번인 사원의 연봉 오름차순으로 출력하시오.
462 SELECT empno, ename, sal, comm, sal * 12 + IFNULL(comm, 0) AS "Annual"
463 FROM emp WHERE deptno = 20
464 ORDER BY "Annual" ASC;
465
466 --부서번호로 정렬한 후, 부서번호가 같을 경우 급여가 많은 순으로 사원번호, 사원이름, 업무, 부서번호, 급여를 출력하시오.
467 SELECT empno, ename, job, deptno, sal
468 FROM emp
469 ORDER BY deptno ASC, sal DESC;
```