

知识点列表

编号	名称	描述	级别
1	单行函数的使用	在查询语句中使用函数实现功能	**
2	排序	学会将查询结果排序	***
3	组函数的使用	在查询语句中使用函数实现功能	***
4	数据字典	了解数据字典的概念及功能	*
5	子查询	在查询语句的条件中使用另一查询语句的结果	***

注： **"理解级别 ***"掌握级别 ****"应用级别

目录

1. SQL*Plus 简介 *	4
2. 查询语句中使用函数	4
2.1. 数字函数	4
2.1.1. round() **	4
2.1.2. 数字函数 trunc() **	5
2.2. 日期函数 **	5
2.2.1. 日期函数 sysdate	6
2.2.2. 虚表 dual **	6
2.2.3. 日期数据相减 **	7
2.2.4. 日期函数 months_between() **	8
2.2.5. 日期函数 add_months() *	9
2.2.6. 日期函数 last_day () *	9
2.3. 转换函数	9
2.3.1. 转换函数 to_char(日期数据 , 格式): 把日期数据转换为字符数据	10
2.3.2. 日期格式 *	11
2.3.3. 转换函数 to_date() **	11
2.3.4. 小结 : 转换函数 to_date()和 to_char() **	13
2.4. 其他函数	13
2.4.1. 函数 coalesce() **	13
2.4.2. case 语句 **	14
2.4.3. decode 函数 **	15
2.5. 函数的嵌套 **	16
2.6. 查询结果排序 order by **	17

2.7. select 语句示意图.....	20
2.8. 组函数	20
2.8.1. 数据字典 user_tables **	20
2.8.2. Sqlplus 命令 **	22
2.9. 组函数 count () avg() sum() max() min() **	23
2.10. 分组查询 group by **	25
2.11. having 子句 **	28
3. 子查询(一) **	30

1. SQL*Plus 简介 *

Oracle 的 sql*plus 是与 oracle 数据库进行交互的命令行工具。它功能强大，使用简单，可以单独安装，也可以和 Oracle 数据库一起安装。

在 sql*plus 中，可以运行 sql 命令与 sql*plus 语句。

我们通常所说的 DML、DDL、DCL 语句都是 **sql 语句**，它们执行完后，都可以保存在一个被称为 sql buffer 的内存区域中，并且只能保存一条最近执行的 sql 语句，我们可以对保存在 sql buffer 中的 sql 语句进行修改，然后再次执行。

除了 sql 语句，在 sql*plus 中执行的其它语句我们称之为 **sql*plus 命令**。它们执行完后，不保存在 sql buffer 的内存区域中，它们一般用来对输出的结果进行格式化显示，以便于制作报表等。

2. 查询语句中使用函数

2.1. 数字函数

2.1.1. round() **

round(数字, 小数点后的位数)用于数字的**四舍五入**

【案例 1】计算金额的四舍五入

```
SQL> select ename, salary * 0.1234567 s1,      --原样显示
        round( salary * 0.1234567, 2 ) s2,      --保留 2 位有效数字
        round( salary * 0.1234567 ) s3,        --默认 0 位有效数字
        from emp_xxx ;
```

ENAME	S1	S2	S3
张无忌	1234. 567	1234. 57	1235
刘苍松	987. 6536	987. 65	988
李翊	1111. 1103	1111. 11	1111
郭芙蓉	617. 2835	617. 28	617
张三丰	1851. 8505	1851. 85	1852

陆无双	370.3701	370.37	370
黄蓉	617.2835	617.28	617
韦小宝	493.8268	493.83	494
郭靖	555.55515	555.56	556
燕小六	617.2835	617.28	617
余泽成			

2.1.2. 数字函数 trunc() **

trunc(数字, 小数点后的位数)用于**截取**

✓ 如果没有第二个参数, 默认是 0

【案例 2】计算金额, 末尾不做四舍五入

```
SQL> select ename, salary * 0.1234567 s1,
        round( salary * 0.1234567, 2 ) s2,
        round( salary * 0.1234567 ) s3,
        trunc( salary * 0.1234567, 2 ) s4
from emp_xxx;
```

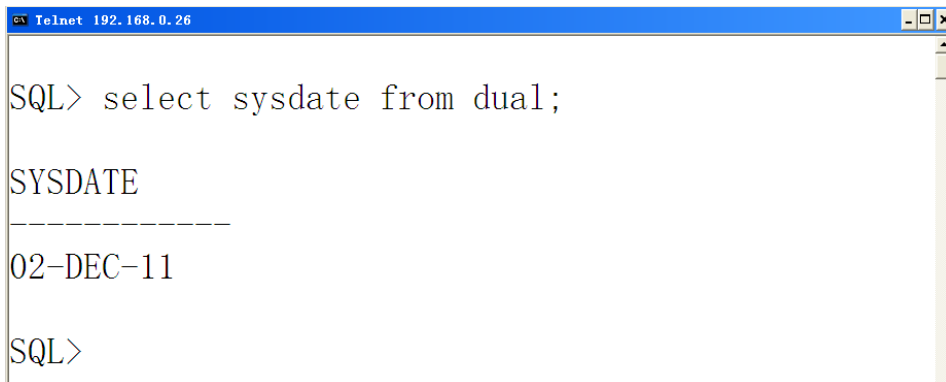
SA Telnet 192.168.0.26				
	S1	S2	S3	S4
	1234.567	1234.57	1235	1234.56
	987.6536	987.65	988	987.65
	1111.1103	1111.11	1111	1111.11
	617.2835	617.28	617	617.28
	1851.8505	1851.85	1852	1851.85
	370.3701	370.37	370	370.37
	617.2835	617.28	617	617.28
	493.8268	493.83	494	493.82
	555.55515	555.56	556	555.55
	617.2835	617.28	617	617.28

2.2. 日期函数 **

2.2.1. 日期函数 sysdate

【案例 3】获取系统当前时间

SQL> select sysdate from dual; --dual 为虚表



```

Telnet 192.168.0.26

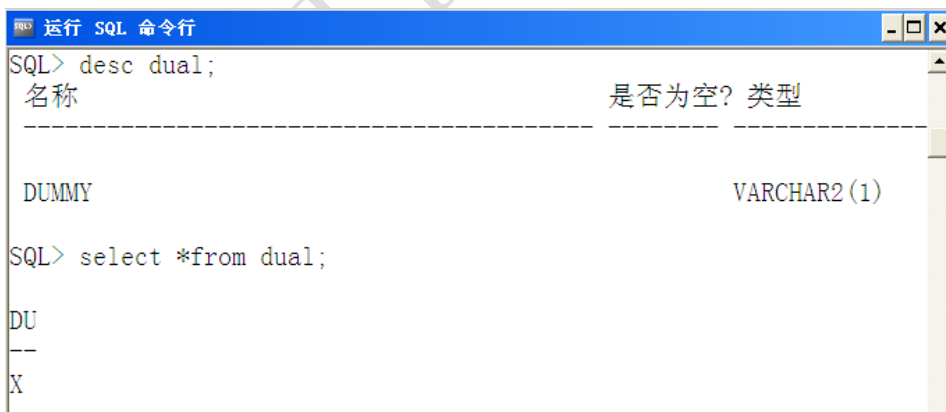
SQL> select sysdate from dual;

SYSDATE
-----
02-DEC-11

SQL>
  
```

2.2.2. 虚表 dual **

虚表 dual 是 Oracle 提供的用于操作函数的方式。属于 SYS 用户，共享给所有用户使用。
虚表 dual 是单行单列的表，表中存放一个常量数据 X。



```

运行 SQL 命令行

SQL> desc dual;
 名称                                是否为空? 类型
-----
DUMMY                                VARCHAR2(1)

SQL> select *from dual;

DU
--
X
  
```

- **虚表的意义**

更方便的操作函数或者查询常量

如下例：表中有多少条数据，常量就会出现多少次

```
运行 SQL 命令行

SQL> select sysdate from dept_xxx;

SYSDATE
-----
05-12月-11
05-12月-11
05-12月-11
05-12月-11
```

2.2.3. 日期数据相减 **

【案例 4】计算员工入职多少天?

```
SQL> select ename , hiredate , ( sysdate - hiredate ) days
      from emp_xxx ;
```

```
运行 SQL 命令行

SQL> select ename, hiredate, (sysdate - hiredate) days
       2  from emp_xxx;
```

ENAME	HIREDATE	DAYS
amy	05-12月-11	.021435185
张无忌	12-5月 -10	572.495752
刘苍松	01-4月 -11	248.495752
李翊	11-4月 -10	603.495752
郭芙蓉	01-1月 -11	338.495752
张三丰	15-5月 -08	1299.49575
燕小六	01-2月 -09	1037.49575
陆无双	01-2月 -09	1037.49575
黄蓉	01-5月 -09	948.495752
韦小宝	20-2月 -09	1018.49575
郭靖	10-5月 -09	939.495752

已选择11行。

- ✓ 日期数据相减，得到两个日期之间的天数差，不足一天用**小数表示**。可以用 round 函数处理一下。

```
SQL> select ename , hiredate , round( sysdate - hiredate ) days
      from emp ;
```

```
运行 SQL 命令行
SQL> select ename, hiredate, round(sysdate - hiredate) days
2 from emp_xxx;
```

ENAME	HIREDATE	DAYS
amy	05-12月-11	0
张无忌	12-5月 -10	573
刘苍松	01-4月 -11	249
李翊	11-4月 -10	604
郭芙蓉	01-1月 -11	339
张三丰	15-5月 -08	1300
燕小六	01-2月 -09	1038
陆无双	01-2月 -09	1038
黄蓉	01-5月 -09	949
韦小宝	20-2月 -09	1019
郭靖	10-5月 -09	940

已选择11行。

2.2.4. 日期函数 months_between() **

【案例 5】 计算员工入职多少个月？小数。

```
SQL> select ename , hiredate ,
           months_between( sysdate , hiredate ) months
           from emp_xxx ;
```

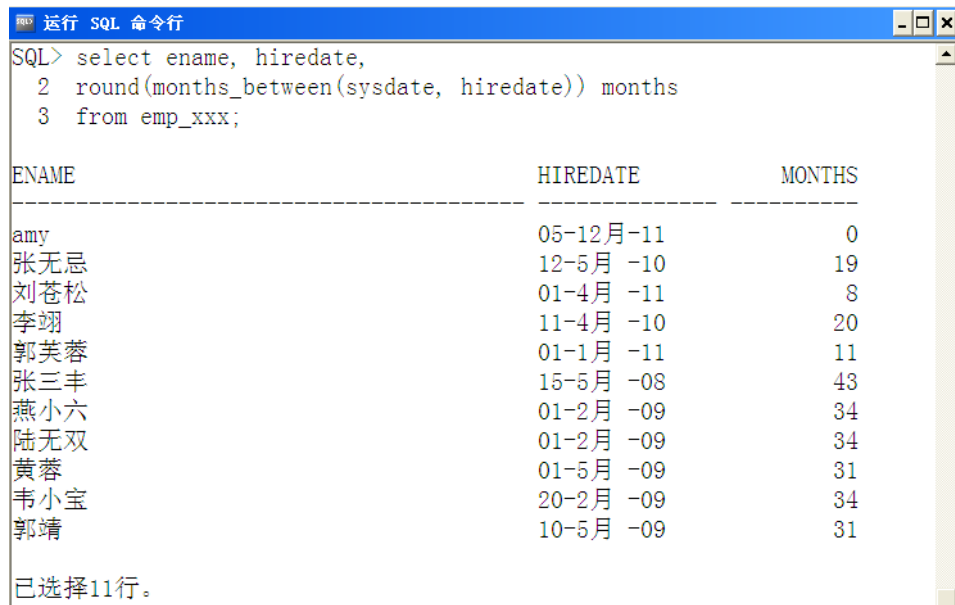
```
运行 SQL 命令行
SQL> select ename, hiredate,
2 months_between(sysdate, hiredate) months
3 from emp_xxx;
```

ENAME	HIREDATE	MONTHS
amy	05-12月-11	0
张无忌	12-5月 -10	18.7906299
刘苍松	01-4月 -11	8.14546856
李翊	11-4月 -10	19.8228879
郭芙蓉	01-1月 -11	11.1454686
张三丰	15-5月 -08	42.6938557
燕小六	01-2月 -09	34.1454686
陆无双	01-2月 -09	34.1454686
黄蓉	01-5月 -09	31.1454686
韦小宝	20-2月 -09	33.5325653
郭靖	10-5月 -09	30.855146

已选择11行。

【案例 6】计算员工入职多少个月？用整数表示

```
SQL> select ename , hiredate ,
        round( months_between( sysdate , hiredate ) ) months
        from emp_xxx ;
```



The screenshot shows a SQL command window titled "运行 SQL 命令行". The query executed is:

SQL> select ename, hiredate,

2 round(months_between(sysdate, hiredate)) months

3 from emp_xxx;

The result is a table with three columns: ENAME, HIREDATE, and MONTHS. The data is as follows:

ENAME	HIREDATE	MONTHS
amy	05-12月-11	0
张无忌	12-5月 -10	19
刘苍松	01-4月 -11	8
李翊	11-4月 -10	20
郭芙蓉	01-1月 -11	11
张三丰	15-5月 -08	43
燕小六	01-2月 -09	34
陆无双	01-2月 -09	34
黄蓉	01-5月 -09	31
韦小宝	20-2月 -09	34
郭靖	10-5月 -09	31

 The window also shows the message "已选择11行。" (11 rows selected).

2.2.5. 日期函数 add_months() *

【案例 7】计算 12 个月之前的时间点

```
SQL> select add_months(sysdate, -12) from dual;
```

2.2.6. 日期函数 last_day () *

【案例 8】计算本月的最后一天

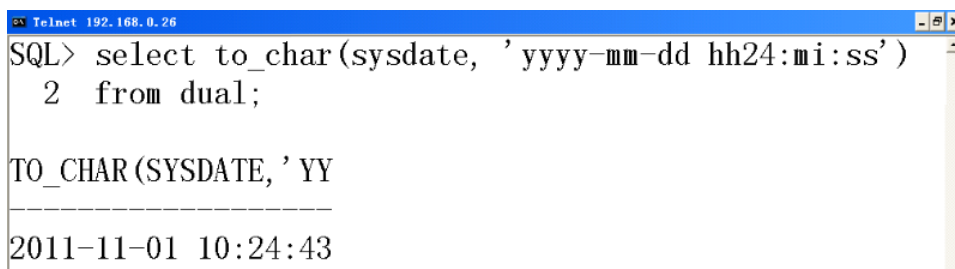
```
SQL> select last_day(sysdate) from dual;
```

2.3. 转换函数

2.3.1. 转换函数 to_char(日期数据, 格式): 把日期数据转换为字符数据

【案例 9】把时间数据按指定格式输出

```
SQL> select to_char( sysdate , 'yyyy-mm-dd hh24 : mi : ss ' )
      from dual ;
```

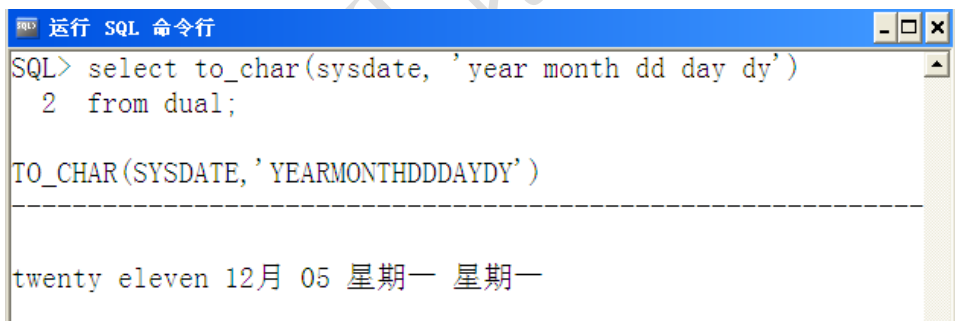


```
SQL> select to_char(sysdate, 'yyyy-mm-dd hh24:mi:ss')
      2  from dual;

TO_CHAR(SYSDATE, 'YY
-----
2011-11-01 10:24:43
```

【案例 10】把时间数据按指定格式输出

```
SQL> select to_char( sysdate , 'year month dd day dy ' )
      from dual ;
```



```
SQL> select to_char(sysdate, 'year month dd day dy')
      2  from dual;

TO_CHAR(SYSDATE, 'YEARMONTHDDDAYDY')
-----
twenty eleven 12月 05 星期一 星期一
```

【案例 11】把时间数据按指定格式输出

```
SQL> select to_char( sysdate , 'yyyy/mm/dd' ) from dual ;
```

```

运行 SQL 命令行
SQL> select to_char(sysdate, 'yyyy/mm/dd')
      2 from dual;

TO_CHAR(SYSDATE,'YYY
-----
2011/12/05

SQL>

```

2.3.2. 日期格式 *

1) 常用日期格式

- ✓ yyyy 四位数字年 如：2011
- ✓ year 全拼的年 如：twenty eleven
- ✓ month 全拼的月 如：november 或 11 月(中文)
- ✓ mm 两位数字月 如：11
- ✓ mon 简拼的月 如：nov(中文没有简拼)
- ✓ dd 两位数字日
- ✓ day 全拼的星期 如：tuesday
- ✓ dy 简拼的星期 如：tue
- ✓ am 上午/下午 如：am/pm

2) sqlplus 中日期的默认格式是：DD-MON-RR 日-月-年

年(RR)和年(YY)的区别：

YY 格式是 Oracle 早期的时间表示方式，存在**千年虫问题**，如下所示

- 假设现在时间为 2011 年，05 年和 98 年分别被 Oracle 解释为：

	YY	RR
05 年	2005	2005
98 年	2098	1998

- 假设现在时间为 1998 年，05 年和 98 年分别被 Oracle 解释为：

	YY	RR
05 年	1905	2005
95 年	1995	1995

2.3.3. 转换函数 to_date() **

【案例 12】插入一条数据，编号为 1012，姓名为 amy，入职时间为当前系统时间

```
SQL> insert into emp_xxx( empno ,ename ,hiredate )  
      values( 1012 , 'amy' , sysdate );
```

【案例 13】插入一条数据，编号为 1012，姓名为 amy，入职时间为 2011 年 10 月 10 日

```
SQL> insert into emp_xxx( empno ,ename ,hiredate )  
      values( 1012 , 'amy' , '10-OCT-11 ' );  
      --入职时间是 2011-10-10
```

✓ 这种时间格式**不符合**中国常用时间表示格式

【案例 14】按指定时间格式插入数据

```
SQL> insert into emp_xxx( empno ,ename ,hiredate )  
      values( 1012 , 'amy' ,  
            to_date( '2011-10-10' , 'yyyy-mm-dd' ) );  
  
SQL> select * from emp_xxx where empno = 1012;
```

【案例 15】按指定格式显示员工姓名和入职时间，显示格式为： amy 2011-10-10

```
SQL> select ename , to_char(hiredate , 'yyyy-mm-dd' ) from emp_xxx ;
```

```
运行 SQL 命令行
SQL> select ename,to_char(hiredate,'yyyy-mm-dd') from emp_xxx;

ENAME                                TO_CHAR(HIREDATE,'YY
-----
amy                                  2011-12-05
张无忌                             2010-05-12
刘苍松                             2011-04-01
李翊                               2010-04-11
郭芙蓉                             2011-01-01
张三丰                             2008-05-15
燕小六                             2009-02-01
陆无双                             2009-02-01
黄蓉                               2009-05-01
韦小宝                             2009-02-20
郭靖                               2009-05-10

已选择11行。
```

2.3.4. 小结：转换函数 to_date()和 to_char() **

to_date()和 to_char()是时间处理的函数

- ✓ to_date 将字符串数据 按指定格式 转换为 日期数据
- ✓ to_char 将日期数据 按指定格式 转换为 字符串数据

2.4. 其他函数

2.4.1. 函数 coalesce() **

coalesce(参数列表)函数的作用：

- ✓ 返回参数列表中第一个非空参数，参数列表中最后一个值通常为常量

【案例 16】计算员工的年终奖金

要求：

- 1) 如果 bonus 不是 null，发年终奖金额为 bonus
- 2) 如果 bonus 是 null，发年终奖金额为 salary * 0.5
- 3) 如果 bonus 和 salary 都是 null, 发 100 元安慰一下

```
SQL> select ename , bonus , salary ,
           coalesce( bonus , salary*0.5 , 100 ) bonus
```

from emp_xxx ;

--返回参数列表中第一个非空数据

--最后一个参数通常是常量

```
运行 SQL 命令行
SQL> select ename, bonus, salary,
2 coalesce(bonus, salary*0.5, 100) bonus
3 from emp_xxx;
```

ENAME	BONUS	SALARY	BONUS
amy			100
张无忌	2000	10000	2000
刘苍松	1000	8000	1000
李翊	1000	9000	1000
郭芙蓉		5000	2500
张三丰		15000	7500
燕小六	400	5000	400
陆无双	500	3000	500
黄蓉	500	5000	500
韦小宝		4000	2000
郭靖	500	4500	500

已选择11行。

2.4.2. case 语句 **

case 语句是数据中的分支语句，相当于 Java 中的 switch-case 语句。

【案例 17】根据员工的职位，计算加薪后的薪水数据

要求：

- 1) 如果职位是 Analyst：加薪 10%
- 2) 如果职位是 Programmer：加薪 5%
- 3) 如果职位是 clerk：加薪 2%
- 4) 其他职位：薪水不变

```
SQL> select ename , salary , job ,
      case job when 'Analyst' then salary * 1.1 --注意这里没有 " , "
            when 'Programmer' then salary * 1.05
            when 'clerk' then salary * 1.02
            else salary --else 相当于 Java 中 case 语句的 default
      end new_salary --end 是 case 语句的结束标识
```

```

from emp_xxx;          --new_salary 是从 case 开始到 end 结束这部分
                        的别名
运行 SQL 命令行
SQL> select ename, salary, job,
2 case job when 'Analyst' then salary * 1.1
3         when 'Programmer' then salary * 1.05
4         when 'clerk' then salary * 1.02
5 else salary
6 end new_salary
7 from emp_xxx;

ENAME                SALARY JOB                NEW_SALARY
-----
amy
张无忌                10000 Manager                10000
刘苍松                8000 Analyst                8800
李翊                 9000 Analyst                9900
郭芙蓉                5000 Programmer            5250
张三丰               15000 President            15000
燕小六                5000 Manager                5000
陆无双                3000 clerk                3060
黄蓉                 5000 Manager                5000
韦小宝                4000 salesman             4000
郭靖                 4500 salesman             4500

已选择11行。

```

2.4.3. decode 函数 **

decode()函数是 Oracle 中等价于 case when 语句的函数，作用同 case 语句相同。

decode 函数语法如下：

decode(判断条件, 匹配 1, 值 1, 匹配 2, 值 2, ... , 默认值)

表达的意思是：如果判断条件 = 匹配 1，则返回值 1

判断条件 = 匹配 2，则返回值 2

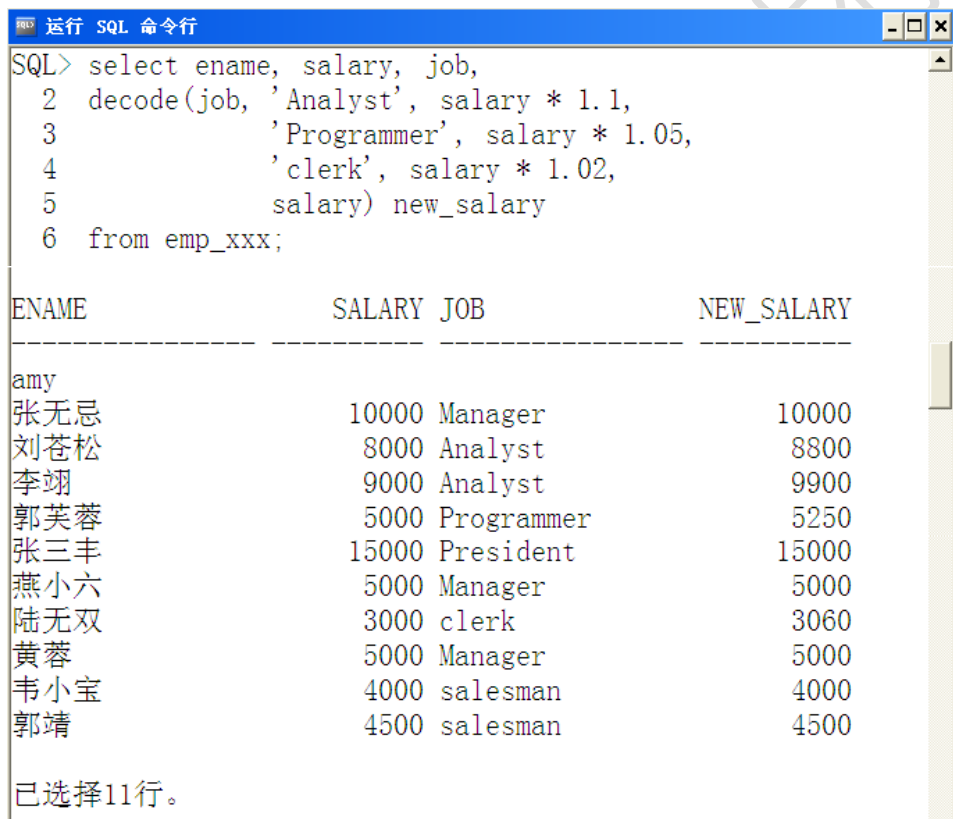
【案例 18】根据员工的职位，计算加薪后的薪水数据

要求：和 case 语句相同

- 1) 如果职位是 Analyst：加薪 10%
- 2) 如果职位是 Programmer：加薪 5%
- 3) 如果职位是 clerk：加薪 2%
- 4) 其他职位：薪水不变

```
SQL> select ename , salary , job ,
        decode(job , 'Analyst' , salary * 1.1 ,
                'Programmer' , salary * 1.05 ,
                'clerk' , salary * 1.02 ,
                salary)
        new_salary
        from emp_xxx ;
```

- ✓ 如果 job 为 Analyst 则 salary*1.1 ;如果为 Programmer 则 salary*1.05 ;如果为 clerk 则 salary*1.02 ;否则值为 salary(不变)



The screenshot shows a window titled "运行 SQL 命令行" (Run SQL Command Line). It contains the same SQL query as above. Below the query, the results are displayed in a table format with columns: ENAME, SALARY, JOB, and NEW_SALARY. The results show 11 rows of data, including employees like amy, 张无忌, 刘苍松, etc. At the bottom, it says "已选择11行。" (11 rows selected).

ENAME	SALARY	JOB	NEW_SALARY
amy	10000	Manager	10000
张无忌	8000	Analyst	8800
刘苍松	9000	Analyst	9900
李翊	5000	Programmer	5250
郭芙蓉	15000	President	15000
张三丰	5000	Manager	5000
燕小六	3000	clerk	3060
陆无双	5000	Manager	5000
黄蓉	4000	salesman	4000
韦小宝	4500	salesman	4500
郭靖			

已选择11行。

2.5. 函数的嵌套 **

函数的嵌套 : f3(f2(f1(p1,p2) , p3) , p4)

2.6. 查询结果排序 order by **

【案例 19】薪水由低到高排序(升序排列)

```
SQL> select ename , salary from emp_xxx
      order by salary asc ;           --正序排列 , asc 可以省略
```

运行 SQL 命令行

```
SQL> select ename, salary from emp_xxx
      2  order by salary;
```

ENAME	SALARY
陆无双	3000
韦小宝	4000
郭靖	4500
郭芙蓉	5000
燕小六	5000
黄蓉	5000
刘苍松	8000
李翊	9000
张无忌	10000
张三丰	15000
amy	

已选择11行。

✓ 空值被看做最大

【案例 20】薪水由高到低排序(降序排列)

```
SQL> select ename, salary from emp_xxx
      order by salary desc ;         --desc( descend )降序排列 不可省略
```

```
运行 SQL 命令行
SQL> select ename, salary from emp_xxx
2 order by salary desc;

ENAME          SALARY
-----
amy
张三丰          15000
张无忌          10000
李翊            9000
刘苍松          8000
郭芙蓉          5000
燕小六          5000
黄蓉            5000
郭靖            4500
韦小宝          4000
陆无双          3000

已选择11行。
```

【案例 21】按入职时间排序，入职时间越早排在前面

```
SQL> select ename, hiredate
      from emp_xxx
      order by hiredate;
```

```
运行 SQL 命令行
SQL> select ename, hiredate
2  from emp_xxx
3  order by hiredate;

ENAME          HIREDATE
-----
张三丰          15-5月 -08
陆无双          01-2月 -09
燕小六          01-2月 -09
韦小宝          20-2月 -09
```

黄蓉	01-5月 -09
郭靖	10-5月 -09
李翊	11-4月 -10
张无忌	12-5月 -10
郭芙蓉	01-1月 -11
刘苍松	01-4月 -11
amy	05-12月-11

已选择11行。

【案例 22】按部门排序，同一部门按薪水由高到低排序

```
SQL> select ename , deptno , salary
      from emp_xxx
      order by deptno , salary desc ;
```

运行 SQL 命令行

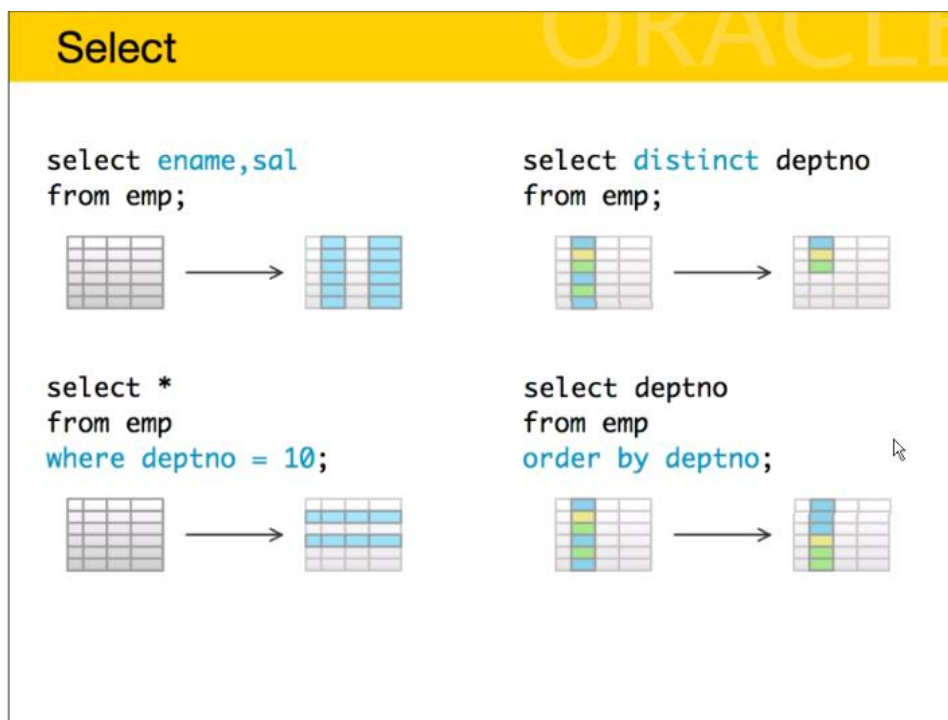
```
SQL> select ename, deptno, salary
2  from emp_xxx
3  order by deptno, salary desc;
```

ENAME	DEPTNO	SALARY
张无忌	10	10000
李翊	10	9000
刘苍松	10	8000
郭芙蓉	10	5000
张三丰	20	15000
燕小六	20	5000
陆无双	20	3000
黄蓉	30	5000
郭靖	30	4500
韦小宝	30	4000
amy		

已选择11行。

注意：排序语句放在查询语句的最后；

2.7. select 语句示意图

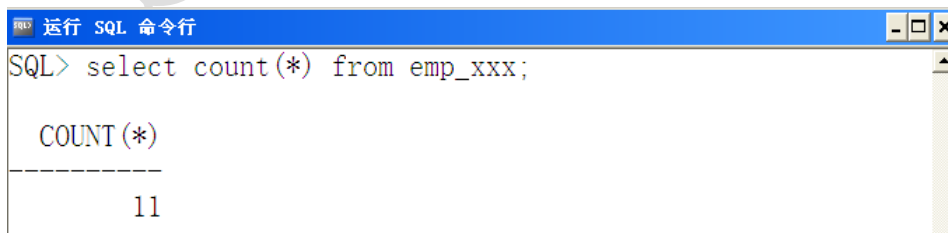


- ✓ 排序可以用列名，列别名，表达式，函数，甚至可以用数字，表示按第几列排序

2.8. 组函数

【案例 23】员工表中有多少条记录？

```
SQL> select count( * ) from emp_xxx ;
```



- ✓ Emp_xxx 表中共有 11 条数据

2.8.1. 数据字典 user_tables **

user_tables 只读，不能改。

【案例 24】当前帐户(openlab)下有多少个表？

```
SQL> select count(*) from user_tables ;
```

```
运行 SQL 命令行
SQL> select count(*) from user_tables;

COUNT(*)
-----
        671
```

```
运行 SQL 命令行
SQL> desc user_tables;

名称                                     是否为空? 类型
-----
TABLE_NAME                             NOT NULL VARCHAR2
TABLESPACE_NAME                         VARCHAR2
CLUSTER_NAME                            VARCHAR2
IOT_NAME                                VARCHAR2
STATUS                                  VARCHAR2
PCT_FREE                                NUMBER
PCT_USED                                NUMBER
INI_TRANS                                NUMBER
MAX_TRANS                                NUMBER
INITIAL_EXTENT                           NUMBER
NEXT_EXTENT                              NUMBER
MIN_EXTENTS                              NUMBER
MAX_EXTENTS                              NUMBER
```

【案例 25】openlab 帐户下有多少个名字中包含 emp 的表？

```
SQL> select count(*) from user_tables
      where table_name like '%EMP%';
```

-- **注意**：表名默认全部大写

-- like 是模糊查询，%代表 0 到多个字符

```
运行 SQL 命令行
SQL> select count(*) from user_tables
2 where table_name like '%EMP%';

COUNT(*)
-----
4
```

【案例 26】入职时间不是 null 的数据总数

SQL> select count(hiredate) from emp_xxx; --注意：count 函数忽略空值

```
运行 SQL 命令行
SQL> select ename , hiredate from emp_xxx;

ENAME          HIREDATE
-----
amy            05-12月-11
张无忌         12-5月 -10
刘苍松         01-4月 -11
李翊           11-4月 -10
郭芙蓉         01-1月 -11
张三丰         15-5月 -08
燕小六         01-2月 -09
陆无双         01-2月 -09
黄蓉           01-5月 -09
韦小宝         20-2月 -09
郭靖           10-5月 -09
余泽成         

已选择12行。

SQL> select count(hiredate) from emp_xxx;

COUNT (HIREDATE)
-----
11
```

2.8.2. Sqlplus 命令 **

- 查看当前用户账户 show user

```

运行 SQL 命令行
SQL> show user;
USER 为 "SYS"
SQL> select user from dual;

USER
-----
SYS
    
```

✓ show user 和 select user from dual 语句作用相同

2.9. 组函数 count() avg() sum() max() min() **

与单行函数如 round()、to_date()、to_char()、coalesce()等不同，单行函数是每行数据返回一行结果，组函数是多行数据返回一行结果。

【案例 27】计算员工的薪水总和是多少？

```
SQL> select sum(salary) from emp_xxx;
```

```

运行 SQL 命令行
SQL> select sum(salary) from emp_xxx;

SUM(SALARY)
-----
68500
    
```

【案例 28】计算员工的人数总和、薪水总和、平均薪水是多少？

错误写法：

```
SQL> select count(*) num, sum(salary) sum_sal, avg(salary) avg_sal,
        from emp_xxx;
```

```
运行 SQL 命令行
SQL> select count(*) num , sum(salary) sum , avg(salary) avg
2   from emp_xxx;

      NUM      SUM      AVG
-----
      12      68500      6850
```

✓ 薪水平均值 = 薪水总和 / 人数总和 $\text{avg(salary)} = \text{sum(salary)} / \text{count(*)}$
而 avg(salary)只按有薪水的员工人数计算平均值。这样得到的数据不够准确。

正确写法：

```
SQL> select count(*) num , sum(salary) sum_sal ,
      avg(nvl(salary , 0)) avg_sal ,
      from emp_xxx ;
```

```
运行 SQL 命令行
SQL> select count(*) num , sum(salary) sum ,
2   avg(nvl(salary,0)) avg
3   from emp_xxx;

      NUM      SUM      AVG
-----
      12      68500  5708.33333
```

【案例 29】计算员工的最高薪水和最低薪水

```
SQL> select max(salary) max_sal , min(salary) min_sal
      from emp_xxx ;
```

```
运行 SQL 命令行
SQL> select max(salary) max_sal,
2   min(salary) min_sal
3   from emp_xxx;

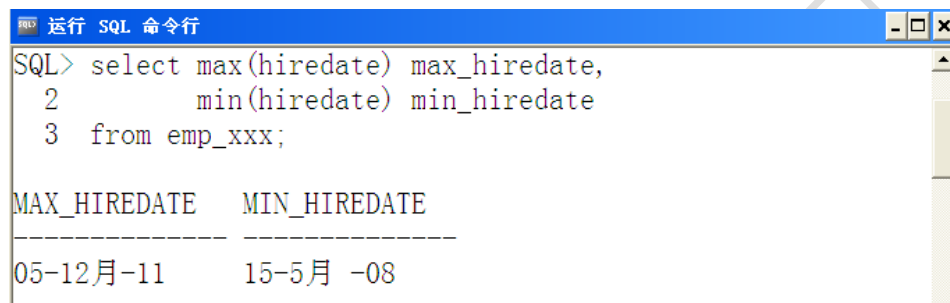
MAX_SAL  MIN_SAL
-----
 15000    3000
```


注意：

- ✓ 组函数：count / avg / sum / max / min 如果函数中写列名，默认忽略空值
- ✓ avg / sum 针对数字的操作
- ✓ max / min 对所有数据类型都可以操作

【案例 30】计算最早和最晚的员工入职时间

```
SQL> select max(hiredate) max_hiredate , min(hiredate) min_hiredate
        from emp_xxx ;
```



```
SQL> select max(hiredate) max_hiredate,
2          min(hiredate) min_hiredate
3          from emp_xxx;

MAX_HIREDATE  MIN_HIREDATE
-----
05-12月-11    15-5月 -08
```

2.10. 分组查询 group by **

group by 列名：表示按指定列分组查询

【案例 31】按部门计算每个部门的最高和最低薪水分别是多少？

需要的数据格式如下：

deptno	max_sal	min_sal
10	4500	3000
20	15000	8000
30	10000	5000

```
SQL> select deptno , max(salary) max_s , min(salary) min_s
        from emp_xxx
        group by deptno ;
```

```
运行 SQL 命令行
SQL> select deptno, max(salary) max_s, min(salary) min_s
2  from emp_xxx
3  group by deptno;
```

DEPTNO	MAX_S	MIN_S
30	5000	4000
20	15000	3000
10	10000	5000

【案例 32】计算每个部门的 薪水总和 和 平均薪水？

```
SQL> select deptno , sum(salary) sum_s , avg(nvl(salary,0)) avg_s
      from emp_xxx
      group by deptno ;
```

```
运行 SQL 命令行
SQL> select deptno, sum(salary) sum_s,
2          avg(nvl(salary,0)) avg_s
3  from emp_xxx
4  group by deptno;
```

DEPTNO	SUM_S	AVG_S
30	13500	4500
20	23000	7666.66667
10	32000	8000

【案例 33】每个部门的统计信息：

要求格式如下：

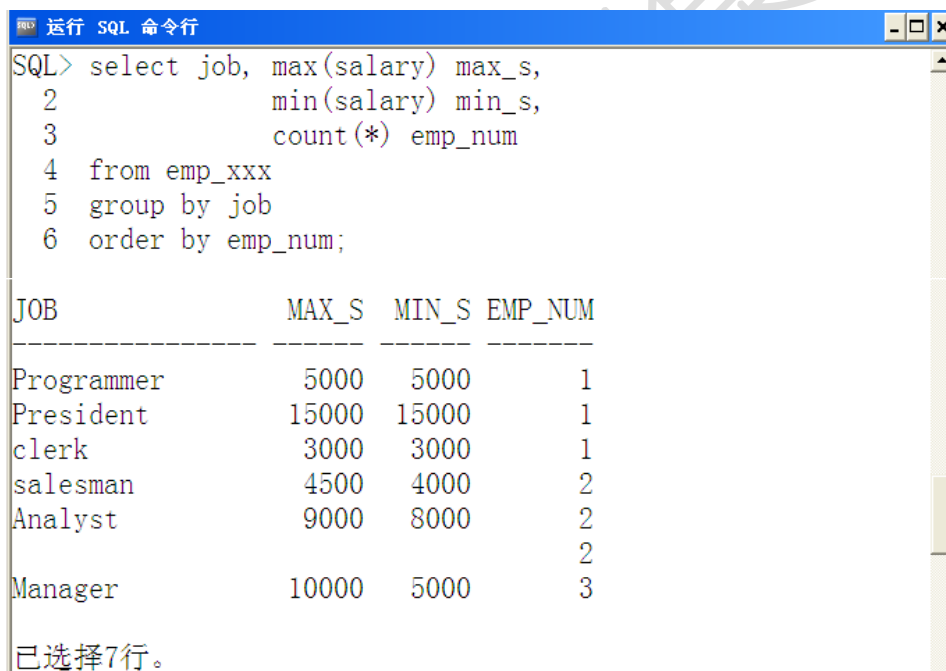
deptno	max_s	min_s	sum_s	avg_s	emp_num
10	10000	5000	23000	6789	3

```
SQL> select deptno, max(salary) max_s ,
      min(salary) min_s ,
      sum(salary) sum_s ,
```

```
avg(nvl(salary, 0)) avg_s ,
count(*) emp_num
from emp_xxx
group by deptno ;
```

【案例 34】按职位分组，每个职位的最高、最低薪水和人数？

```
SQL> select job , max(salary) max_s ,
           min(salary) min_s ,
           count(*) emp_num
from emp_xxx
group by job
order by emp_num ;
```



```
SQL> select job, max(salary) max_s,
2           min(salary) min_s,
3           count(*) emp_num
4 from emp_xxx
5 group by job
6 order by emp_num;
```

JOB	MAX_S	MIN_S	EMP_NUM
Programmer	5000	5000	1
President	15000	15000	1
clerk	3000	3000	1
salesman	4500	4000	2
Analyst	9000	8000	2
Manager	10000	5000	3

已选择7行。

注意：select 后出现的列，凡是没有被组函数包围的列，必须出现在 group by 短语中

【案例 35】如果 select 后没有被组函数的列，没有出现在 group by 短语中，会出错：

第 1 行出现错误:

ORA-00937: 不是单组分组函数

```
SQL> select deptno , max(salary) max_s ,
```

```
min(salary) min_s ,
sum(salary) sum_s ,
avg(nvl(salary, 0)) avg_s ,
count(*) emp_num
from emp_xxx ;
```

【案例 36】如果 group by 短语中的列，没有出现在 select 短语中，不会出错，信息不够全

```
SQL> select max(salary) max_s ,
min(salary) min_s ,
sum(salary) sum_s ,
avg( nvl(salary, 0) ) avg_s ,
count(*) emp_num
from emp_xxx
group by deptno ;
```

查询结果中没有部门信息：

MAX_S	MIN_S	SUM_S	AVG_S	EMP_NUM
-----	-----	-----	-----	-----
5000	4000	13500	4500	3
15000	3000	23000	7666.66667	3
10000	5000	37500	7500	5

2.11. having 子句 **

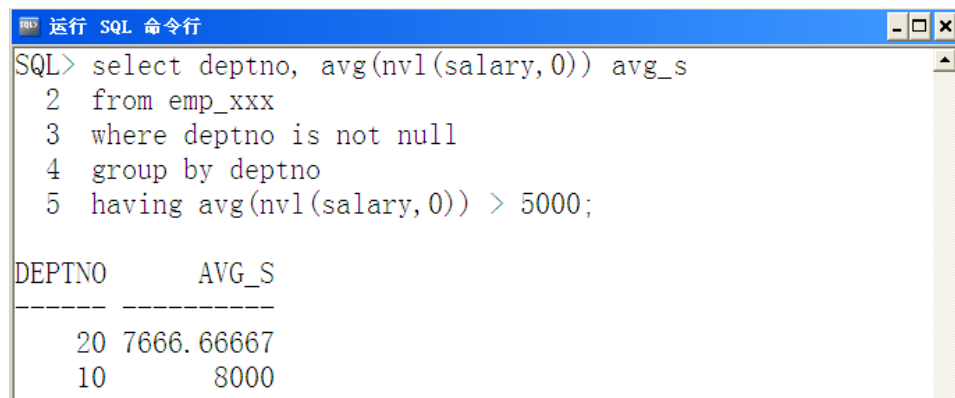
having 子句用于对分组后的数据进行过滤。

注意区别 where 是对表中数据的过滤 ;having 是对分组得到的结果数据进一步过滤

【案例 37】平均薪水大于 5000 元的部门数据，没有部门的不算在内？

```
SQL> select deptno , avg(nvl(salary, 0)) avg_s
from emp_xxx
where deptno is not null
```

```
group by deptno
having avg(nvl(salary, 0)) > 5000;
```

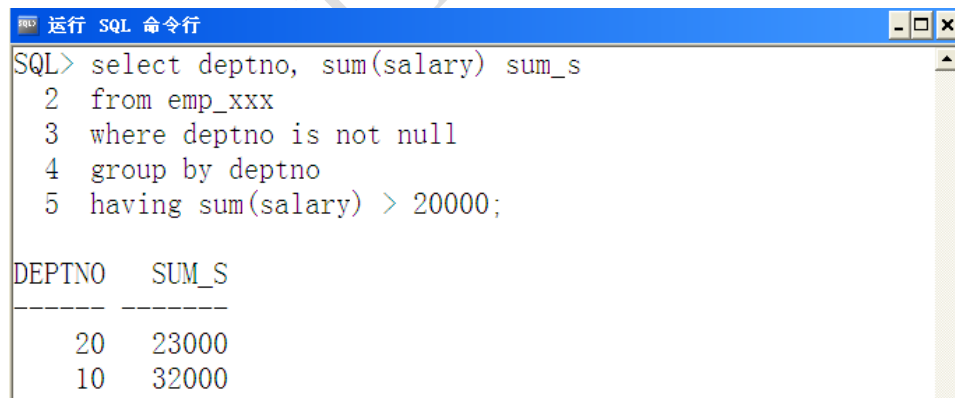


```
SQL> select deptno, avg(nvl(salary,0)) avg_s
2  from emp_xxx
3  where deptno is not null
4  group by deptno
5  having avg(nvl(salary,0)) > 5000;
```

DEPTNO	AVG_S
20	7666.66667
10	8000

【案例 38】薪水总和大于 20000 元的部门数据？

```
SQL> select deptno , sum(salary) sum_s
      from emp_xxx
      where deptno is not null
      group by deptno
      having sum(salary) > 20000 ;
```



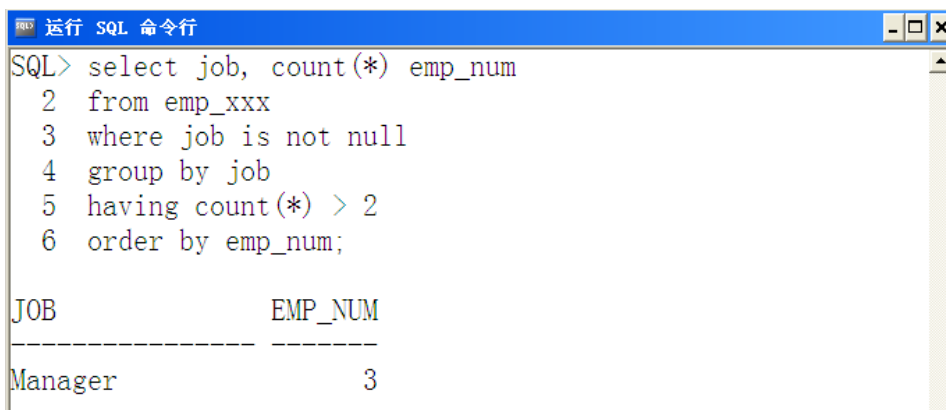
```
SQL> select deptno, sum(salary) sum_s
2  from emp_xxx
3  where deptno is not null
4  group by deptno
5  having sum(salary) > 20000;
```

DEPTNO	SUM_S
20	23000
10	32000

【案例 39】哪些职位的人数超过 2 个人？

```
SQL> select job , count(*) emp_num
      from emp_xxx
      where job is not null
```

```
group by job
having count(*) > 2
order by emp_num ;
```



运行 SQL 命令行

```
SQL> select job, count(*) emp_num
2  from emp_xxx
3  where job is not null
4  group by job
5  having count(*) > 2
6  order by emp_num;
```

JOB	EMP_NUM
Manager	3

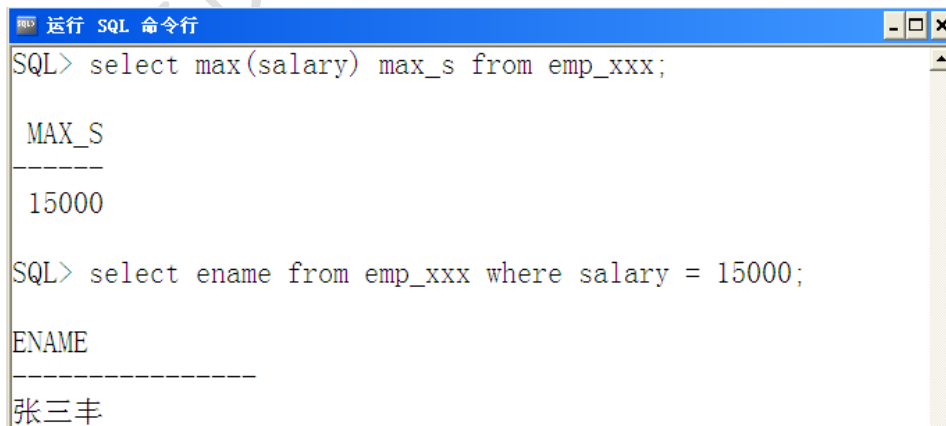
✓ 注意：order by 一定要放到最后

3. 子查询(一)**

【案例 40】查询最高薪水的是谁？

- 分两步：

```
SQL> select max(salary) max_s from emp_xxx ;
SQL> select ename from emp_xxx where salary = 15000 ;
```



运行 SQL 命令行

```
SQL> select max(salary) max_s from emp_xxx;
```

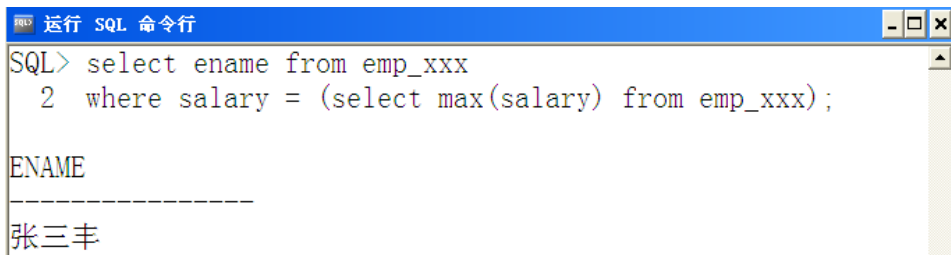
MAX_S
15000

```
SQL> select ename from emp_xxx where salary = 15000;
```

ENAME
张三丰

- 子查询：

```
SQL> select ename from emp_xxx
      where salary = ( select max(salary) from emp_xxx );
```



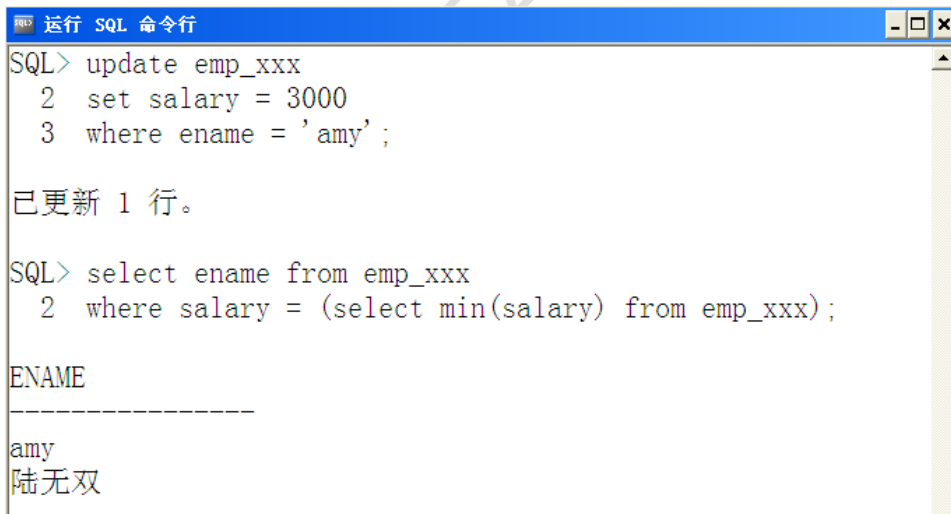
```
运行 SQL 命令行
SQL> select ename from emp_xxx
  2  where salary = (select max(salary) from emp_xxx);

ENAME
-----
张三丰
```

【案例 41】最低薪水的是谁？

```
SQL> update emp_xxx
      set salary = 3000
      where ename = 'amy';  --更新表内容将 amy 的 salary 改为 3000
```

```
SQL> select ename from emp_xxx
      where salary = ( select min(salary) from emp_xxx );
```



```
运行 SQL 命令行
SQL> update emp_xxx
  2  set salary = 3000
  3  where ename = 'amy';

已更新 1 行。

SQL> select ename from emp_xxx
  2  where salary = (select min(salary) from emp_xxx);

ENAME
-----
amy
陆无双
```

- ✓ 查询出最低薪水为 3000 的两名员工