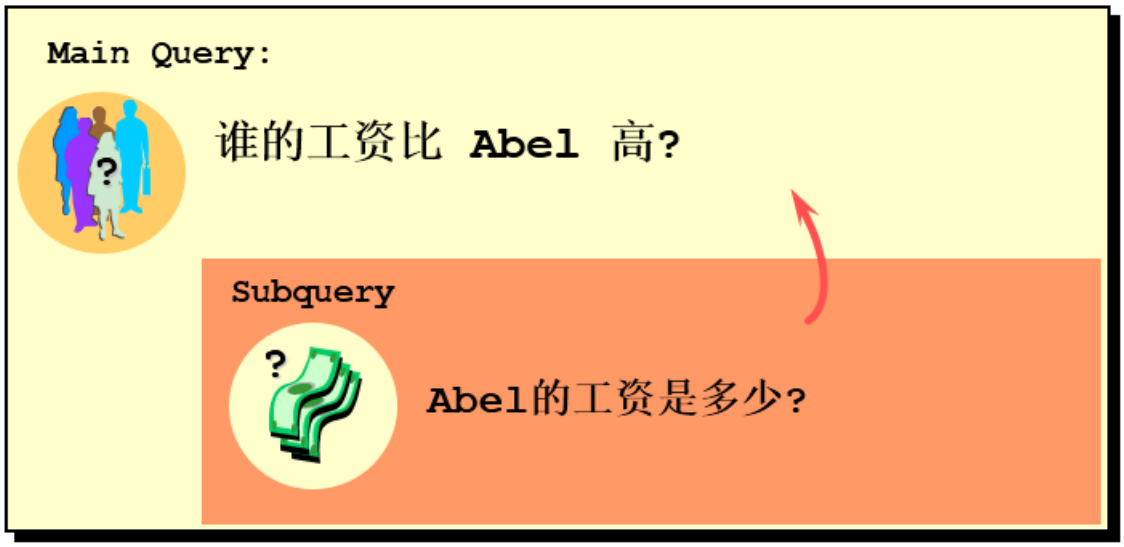
第09章_子查询

子查询指一个查询语句嵌套在另一个查询语句内部的查询

SQL 中子查询的使用大大增强了 SELECT 查询的能力，因为很多时候查询需要从结果集中获取数据，或者需要从同一个表中先计算得出一个数据结果，然后与这个数据结果（可能是某个标量，也可能是某个集合）进行比较。

1. 需求分析与问题解决

1.1 实际问题



现有解决方式：

```
#方式一：
SELECT salary
FROM employees
WHERE last_name = 'Abel';

+----------+
| salary   |
+----------+
| 11000.00 |
+----------+
1 row in set (0.00 sec)

SELECT last_name,salary
FROM employees
WHERE salary > 11000;
```

```
+-----------+----------+
| last_name | salary   |
+-----------+----------+
| King      | 24000.00 |
| Kochhar   | 17000.00 |
| De Haan   | 17000.00 |
| Greenberg | 12000.00 |
| Russell   | 14000.00 |
| Partners  | 13500.00 |
| Errazuriz | 12000.00 |
| Ozer      | 11500.00 |
| Hartstein | 13000.00 |
| Higgins   | 12000.00 |
+-----------+----------+
10 rows in set (0.00 sec)
```

```
#方式二：自连接 + 非等值连接 （刚好可以解决该问题，并不是自连接可以代替子查询）
SELECT e2.last_name,e2.salary
FROM employees e1,employees e2
WHERE e1.last_name = 'Abel'
AND e1.`salary` < e2.`salary`
```

```
#方式三：子查询
SELECT last_name,salary
FROM employees
WHERE salary > (
        SELECT salary
        FROM employees
        WHERE last_name = 'Abel'
        );

+-----------+----------+
| last_name | salary   |
+-----------+----------+
| King      | 24000.00 |
| Kochhar   | 17000.00 |
| De Haan   | 17000.00 |
| Greenberg | 12000.00 |
| Russell   | 14000.00 |
| Partners  | 13500.00 |
| Errazuriz | 12000.00 |
| Ozer      | 11500.00 |
| Hartstein | 13000.00 |
| Higgins   | 12000.00 |
+-----------+----------+
10 rows in set (0.01 sec)
```

称谓规范：外查询（主查询）、内查询（子查询）

1.2 子查询的基本使用

- 子查询的基本语法结构：

```
SELECT    select_list
FROM      table
WHERE     expr operator
                    (SELECT       select_list
                     FROM         table);
```

- 子查询（内查询）在主查询之前一次执行完成。
- 子查询的结果被主查询（外查询）使用 。
- **注意事项**
  - 子查询要包含在括号内
  - 将子查询放在比较条件的**右侧**（可读性）
  - 单行操作符对应单行子查询，多行操作符对应多行子查询

1.3 子查询的分类

分类方式1：

我们按内查询的**结果返回一条还是多条记录**，将子查询分为`单行子查询`、`多行子查询`

- 单行子查询

- 多行子查询

分类方式2：

我们按内查询是**否被执行多次**，将子查询划分为`相关(或关联)子查询`和`不相关(或非关联)子查询`。

不相关子查询：

子查询从数据表中查询了数据结果，如果这个数据结果只执行一次，然后这个数据结果作为主查询的条件进行执行，那么这样的子查询叫做不相关子查询

```
#查询工资比Abel高的人的名字，工资
SELECT last_name,salary
FROM employees
WHERE salary > (
        SELECT salary
        FROM employees
        WHERE last_name = 'Abel'
        );
```

```
# 工资大于公司平均工资的员工信息
```

相关子查询：

先从外部查询开始，每次都传入子查询进行查询，然后再将结果反馈给外部，这种嵌套的执行方式就称为相关子查询

```
-- 查询工资大于本部门平均工资的员工信息
```

2. 单行子查询

2.1 单行比较操作符

操作符	含义
=	equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
<>	not equal to

2.2 代码示例

题目：查询工资大于149号员工工资的员工的信息

```
SELECT last_name,salary
FROM employees
WHERE salary>(
            SELECT salary FROM employees
            WHERE employee_id=149             # 10500.00
            );

+-----------+----------+
| last_name | salary   |
+-----------+----------+
| King      | 24000.00 |
| Kochhar   | 17000.00 |
| De Haan   | 17000.00 |
| Greenberg | 12000.00 |
| Raphaely  | 11000.00 |
| Russell   | 14000.00 |
| Partners  | 13500.00 |
| Errazuriz | 12000.00 |
| Cambrault | 11000.00 |
| Ozer      | 11500.00 |
| Abel      | 11000.00 |
| Hartstein | 13000.00 |
| Higgins   | 12000.00 |
+-----------+----------+
13 rows in set (0.00 sec)
```

题目：返回job_id与141号员工相同，salary比143号员工多的员工姓名，job_id和工资

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 141)
AND    salary >
                (SELECT salary
                 FROM   employees
                 WHERE  employee_id = 143);

+-------------+----------+---------+
| last_name   | job_id   | salary  |
+-------------+----------+---------+
| Nayer       | ST_CLERK | 3200.00 |
| Mikkilineni | ST_CLERK | 2700.00 |
| Bissot      | ST_CLERK | 3300.00 |
| Atkinson    | ST_CLERK | 2800.00 |
| Mallin      | ST_CLERK | 3300.00 |
| Rogers      | ST_CLERK | 2900.00 |
| Ladwig      | ST_CLERK | 3600.00 |
| Stiles      | ST_CLERK | 3200.00 |
| Seo         | ST_CLERK | 2700.00 |
| Rajs        | ST_CLERK | 3500.00 |
| Davies      | ST_CLERK | 3100.00 |
+-------------+----------+---------+
11 rows in set (0.00 sec)
```

题目：返回公司工资最少的员工的last_name,job_id和salary

```
-- 方法一：子查询
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary =
                (SELECT MIN(salary)
                 FROM   employees);

+-----------+----------+---------+
| last_name | job_id   | salary  |
+-----------+----------+---------+
| Olson     | ST_CLERK | 2100.00 |
+-----------+----------+---------+
1 row in set (0.00 sec)
```

```
-- 方法二：排序后取极值
SELECT last_name,job_id,salary
FROM employees
ORDER BY salary
LIMIT 1;

+-----------+----------+---------+
| last_name | job_id   | salary  |
+-----------+----------+---------+
| Olson     | ST_CLERK | 2100.00 |
+-----------+----------+---------+
1 row in set (0.00 sec)
```

题目：查询与141号或174号员工的manager_id和department_id相同的其他员工的employee_id，manager_id，department_id

实现方式1：不成对比较 (没理解)

```
SELECT  employee_id, manager_id, department_id
FROM    employees
WHERE   manager_id IN
          (SELECT  manager_id
                   FROM    employees
                   WHERE   employee_id IN (174,141))
AND     department_id IN
          (SELECT  department_id
                   FROM    employees
                   WHERE   employee_id IN (174,141))
AND employee_id NOT IN(174,141);
```

实现方式2：成对比较

```
SELECT  employee_id, manager_id, department_id
FROM    employees
WHERE  (manager_id, department_id) IN
                     (SELECT manager_id, department_id
                      FROM   employees
                      WHERE  employee_id IN (141,174))
AND employee_id NOT IN (141,174);
```

2.3 HAVING 中的子查询

- 首先执行子查询。

- 向主查询中的HAVING 子句返回结果。

题目：查询最低工资大于50号部门最低工资的部门id和其最低工资

```
-- 单行、不相关子查询
SELECT   department_id, MIN(salary)
FROM     employees
GROUP BY department_id
HAVING   MIN(salary) >
                       (SELECT MIN(salary)
                        FROM   employees
                        WHERE  department_id = 50);

+---------------+-------------+
| department_id | MIN(salary) |
+---------------+-------------+
|          NULL |     7000.00 |
|            10 |     4400.00 |
|            20 |     6000.00 |
|            30 |     2500.00 |
|            40 |     6500.00 |
|            60 |     4200.00 |
|            70 |    10000.00 |
|            80 |     6100.00 |
|            90 |    17000.00 |
|           100 |     6900.00 |
|           110 |     8300.00 |
+---------------+-------------+
11 rows in set (0.00 sec)
```

2.4 CASE中的子查询

在CASE表达式中使用单列子查询：

题目：显式员工的employee_id,last_name和location。其中，若员工department_id与location_id为1800的department_id相同，则location为'Canada'，其余则为'USA'。

```
SELECT employee_id, last_name,      CASE department_id
                                    WHEN
             (SELECT department_id FROM departments WHERE location_id = 1800)

                                    THEN 'Canada' ELSE 'USA'
                                    END location
FROM   employees;

+-------------+-------------+----------+
| employee_id | last_name   | location |
+-------------+-------------+----------+
|         100 | King        | USA      |
|         101 | Kochhar     | USA      |
```

```
|         154 | Cambrault    | USA       |
|         155 | Tuvault      | USA       |
|         206 | Gietz        | USA       |
+-------------+--------------+-----------+
107 rows in set (0.01 sec)
```

2.5 子查询中的空值问题

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  last_name = 'Haas');

Empty set (0.00 sec)
```

> **子查询不返回任何行**
>
> 外查询也不会返回任何行

2.5 非法使用子查询

```
SELECT employee_id, last_name
FROM   employees
WHERE  salary =
                (SELECT   MIN(salary)
                 FROM     employees
                 GROUP BY department_id);

#错误（外查询用了=，子查询有多行数据）
-- ERROR 1242 (21000): Subquery returns more than 1 row
```

> **多行子查询使用单行比较符**

3. 多行子查询

- 也称为集合比较子查询
- **内查询返回多行**
- **使用多行比较操作符　　见下：**

3.1 多行比较操作符

操作符	含义
IN	等于列表中的**任意一个**
ANY	需要和单行比较操作符一起使用，和子查询返回的**某一个**值比较
ALL	需要和单行比较操作符一起使用，和子查询返回的**所有**值比较
SOME	实际上是ANY的别名，作用相同，一般常使用ANY

> 体会 ANY 和 ALL 的区别

3.2 代码示例

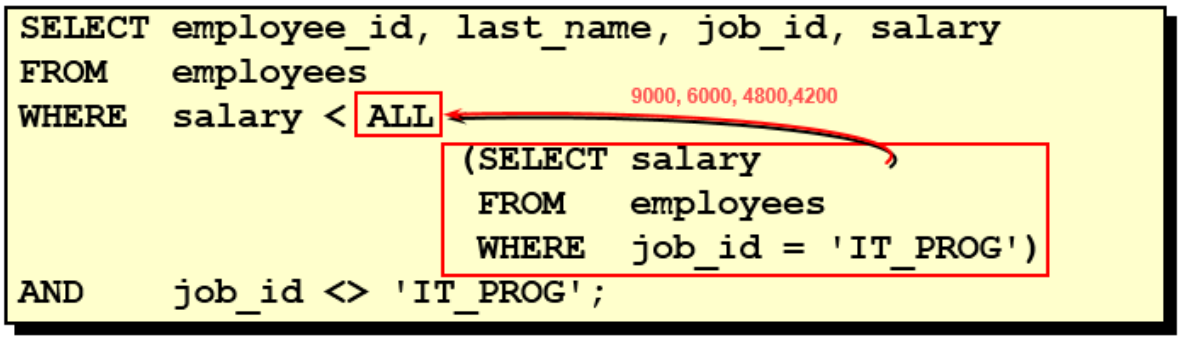
每个部门中工资最低的员工的信息

```
SELECT last_name,salary,department_id
FROM employees
WHERE (department_id,salary) IN(
            SELECT department_id,MIN(salary) FROM employees
            GROUP BY department_id
);

+------------+----------+---------------+
| last_name  | salary   | department_id |
+------------+----------+---------------+
| Kochhar    | 17000.00 |            90 |
| De Haan    | 17000.00 |            90 |
| Lorentz    |  4200.00 |            60 |
| Popp       |  6900.00 |           100 |
| Colmenares |  2500.00 |            30 |
| Olson      |  2100.00 |            50 |
| Kumar      |  6100.00 |            80 |
| Whalen     |  4400.00 |            10 |
| Fay        |  6000.00 |            20 |
| Mavris     |  6500.00 |            40 |
| Baer       | 10000.00 |            70 |
| Gietz      |  8300.00 |           110 |
+------------+----------+---------------+
12 rows in set (0.00 sec)
```

题目：返回其它job_id中比job_id为'IT_PROG'部门任一工资低的员工的员工号、姓名、job_id 以及 salary

```
SELECT last_name,job_id,salary
FROM employees
WHERE job_id <> 'IT_PROG'
AND salary < ANY(
    SELECT salary FROM employees
    WHERE job_id='IT_PROG'
);

+-------------+------------+---------+
| last_name   | job_id     | salary  |
+-------------+------------+---------+
| Chen        | FI_ACCOUNT | 8200.00 |
| Sciarra     | FI_ACCOUNT | 7700.00 |
| Urman       | FI_ACCOUNT | 7800.00 |
| Grant       | SH_CLERK   | 2600.00 |
| Whalen      | AD_ASST    | 4400.00 |
| Fay         | MK_REP     | 6000.00 |
| Mavris      | HR_REP     | 6500.00 |
| Gietz       | AC_ACCOUNT | 8300.00 |
+-------------+------------+---------+
76 rows in set (0.00 sec)
```

题目：返回其它job_id中比job_id为'IT_PROG'部门所有工资都低的员工的员工号、姓名、job_id以及 salary




```
SELECT last_name,job_id,salary
FROM employees
WHERE job_id <> 'IT_PROG'
AND salary < ALL(
    SELECT salary FROM employees
    WHERE job_id='IT_PROG'
```

```
);

+-------------+----------+---------+
| last_name   | job_id   | salary  |
+-------------+----------+---------+
| Khoo        | PU_CLERK | 3100.00 |
| Baida       | PU_CLERK | 2900.00 |
| Tobias      | PU_CLERK | 2800.00 |
| Himuro      | PU_CLERK | 2600.00 |
| Colmenares  | PU_CLERK | 2500.00 |
| Walsh       | SH_CLERK | 3100.00 |
| Feeney      | SH_CLERK | 3000.00 |
| OConnell    | SH_CLERK | 2600.00 |
| Grant       | SH_CLERK | 2600.00 |
+-------------+----------+---------+
44 rows in set (0.00 sec)
```

题目：查询平均工资最低的部门id

```
#方式1：  先获取各个部门的评论工资，再找出最小的平均工资，再用这个最小的平均工资过滤得到目标值
SELECT department_id
FROM employees
GROUP BY department_id
HAVING AVG(salary) = (
            SELECT MIN(avg_sal)
            FROM (
                SELECT AVG(salary) avg_sal
                FROM employees
                GROUP BY department_id
                ) dept_avg_sal #把结果集看成是一张表必须要给表起别名
            );

+---------------+
| department_id |
+---------------+
|            50 |
+---------------+
1 row in set (0.00 sec)
```

对上述的解读：

将SELECT获取到的结果集看作是一个表，放在FROM之后，再进行一轮查询

把结果集看成是一张表必须要给表起别名，否则报错

注意：

MySQL中，聚合函数不能嵌套

Oracle可以

```
SELECT MIN(AVG(salary))
FROM employees
GROUP BY department_id;
-- ERROR 1111 (HY000): Invalid use of group function
```

```
#方式2：平均工资小于等于全部部门的平均工资
SELECT department_id
FROM employees
GROUP BY department_id
HAVING AVG(salary) <= ALL (
                SELECT AVG(salary) avg_sal
                FROM employees
                GROUP BY department_id
);

+---------------+
| department_id |
+---------------+
|            50 |
+---------------+
1 row in set (0.00 sec)
```

3.3 空值问题

```
SELECT last_name
FROM employees
WHERE employee_id NOT IN (
            SELECT manager_id
            FROM employees
            # ORDER BY manager_id DESC # 只要使用NOT IN 且结果中有NULL存在，那么直接结果集为NULL
            );

Empty set (0.00 sec)
```

```
SELECT last_name
FROM employees
WHERE employee_id  IN (
            SELECT manager_id
            FROM employees
            );
```

结论：

只要使用NOT IN 且结果中有NULL存在，那么直接结果集为NULL

注意：

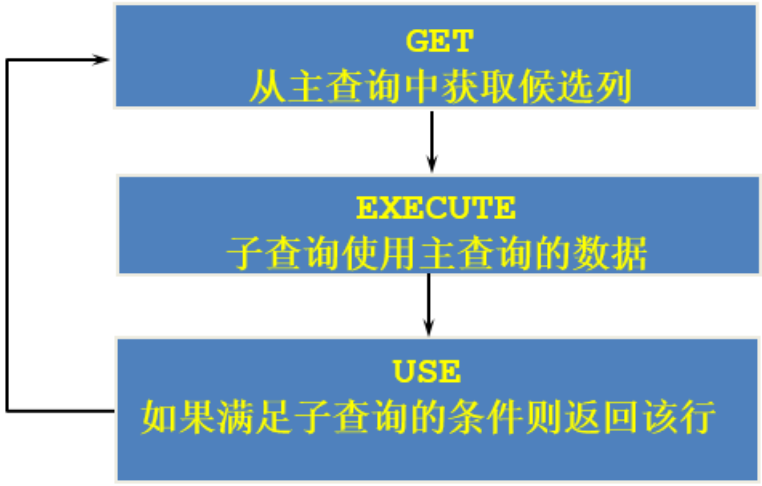
上述都是不相关子查询

4. 相关子查询

4.1 相关子查询执行流程

如果子查询的执行依赖于外部查询，通常情况下都是因为子查询中的表用到了外部的表，并进行了条件关联，因此每执行一次外部查询，子查询都要重新计算一次，这样的子查询就称之为`关联子查询`。

相关子查询按照一行接一行的顺序执行，主查询的每一行都执行一次子查询。




```
SELECT column1, column2, ...
FROM   table1  outer
WHERE  column1 operator
                    (SELECT  colum1, column2
                     FROM    table2
                     WHERE   expr1 =
                                outer.expr2);
```

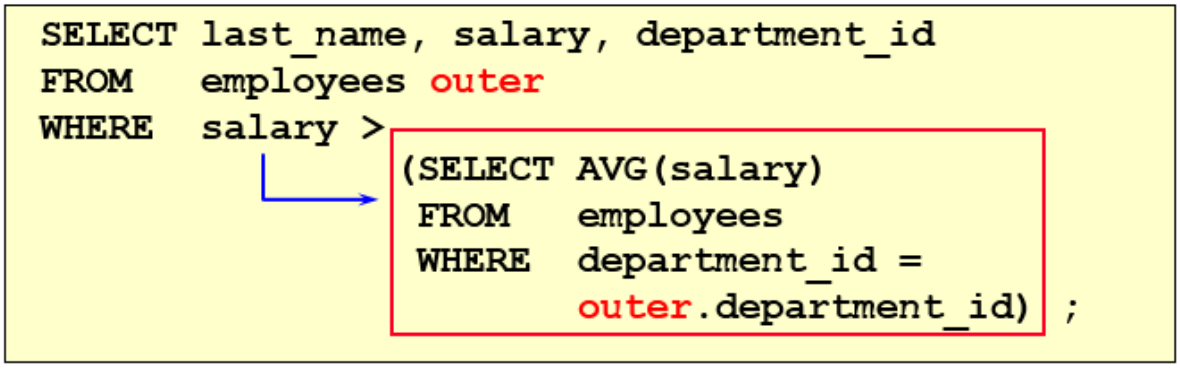
说明：**子查询中使用主查询中的列**

4.2 代码示例

题目：查询员工中工资大于本部门平均工资的员工的last_name,salary和其department_id

方式一：相关子查询

子查询会使用到外查询的一行记录中的某个字段值




```
SELECT last_name,salary
FROM employees e1
WHERE salary >(
    SELECT AVG(salary)
    FROM employees e2
    WHERE e2.department_id= e1.department_id
);

+-----------+----------+
| last_name | salary   |
+-----------+----------+
| King      | 24000.00 |
| Hunold    |  9000.00 |
| Ernst     |  6000.00 |
| Bell      |  4000.00 |
| Everett   |  3900.00 |
| Hartstein | 13000.00 |
| Higgins   | 12000.00 |
+-----------+----------+
38 rows in set (0.00 sec)
```

方式二：在 FROM 中使用子查询

```
SELECT last_name,salary,e1.department_id
FROM employees e1,
(SELECT department_id,AVG(salary) dept_avg_sal FROM employees GROUP BY
department_id) e2
WHERE e1.`department_id` = e2.department_id
AND e2.dept_avg_sal < e1.`salary`;

+-----------+----------+---------------+
| last_name | salary   | department_id |
+-----------+----------+---------------+
| Hartstein | 13000.00 |            20 |
```

```
| King        | 24000.00 |             90 |
| Greenberg   | 12000.00 |            100 |
| Faviet      |  9000.00 |            100 |
| Higgins     | 12000.00 |            110 |
+-------------+----------+----------------+
38 rows in set (0.00 sec)
```

思路：

1、先进行一次查询获取系id与平均工资，将上面的查询当成是一张表；

2、这种查询出来当表的情况记得起别名；

3、然后进行多表查询

> from型的子查询：子查询是作为from的一部分，子查询要用()引起来，并且要给这个子查询取别名，
> 把它当成一张“临时的虚拟的表”来使用

题目：查询员工的id,salary,按照department_name 排序

在ORDER BY 中使用子查询：

```
SELECT employee_id,salary
FROM employees e
ORDER BY (
      SELECT department_name
      FROM departments d
      WHERE e.`department_id` = d.`department_id`
    );

+-------------+----------+
| employee_id | salary   |
+-------------+----------+
|         202 |  6000.00 |
|         204 | 10000.00 |
|         114 | 11000.00 |
|         115 |  3100.00 |
|         197 |  3000.00 |
|         198 |  2600.00 |
|         199 |  2600.00 |
+-------------+----------+
107 rows in set (0.00 sec)
```

题目：若employees表中employee_id与job_history表中employee_id相同的数目不小于2，输出这些相同id的员工的employee_id,last_name和其job_id

先熟悉一下job_history表：

```
SELECT * FROM job_history;

+-------------+------------+------------+------------+---------------+
| employee_id | start_date | end_date   | job_id     | department_id |
+-------------+------------+------------+------------+---------------+
|         101 | 1989-09-21 | 1993-10-27 | AC_ACCOUNT |           110 |
|         101 | 1993-10-28 | 1997-03-15 | AC_MGR     |           110 |
|         102 | 1993-01-13 | 1998-07-24 | IT_PROG    |            60 |
|         114 | 1998-03-24 | 1999-12-31 | ST_CLERK   |            50 |
|         122 | 1999-01-01 | 1999-12-31 | ST_CLERK   |            50 |
|         176 | 1998-03-24 | 1998-12-31 | SA_REP     |            80 |
|         176 | 1999-01-01 | 1999-12-31 | SA_MAN     |            80 |
|         200 | 1987-09-17 | 1993-06-17 | AD_ASST    |            90 |
|         200 | 1994-07-01 | 1998-12-31 | AC_ACCOUNT |            90 |
|         201 | 1996-02-17 | 1999-12-19 | MK_REP     |            20 |
+-------------+------------+------------+------------+---------------+
10 rows in set (0.01 sec)
```

解读题目：查找调岗小于等于两次的员工的 employee_id,last_name和其job_id

```
SELECT e.employee_id, last_name,e.job_id
FROM   employees e
WHERE  2 <= (SELECT COUNT(*)
             FROM   job_history
             WHERE  employee_id = e.employee_id);
```

小结论：

1、在SELECT 中，除了GROUP BY 和 LIMIT 之外；其他位置均可使用子查询；

2、只要发现子查询中出现主查询中的字段就是相关子查询

4.3 EXISTS 与 NOT EXISTS关键字

- **关联子查询通常也会和 EXISTS操作符一起来使用，用来检查在子查询中是否存在满足条件的行**

- **如果在子查询中不存在满足条件的行：**
  - 条件返回 FALSE
  - 继续在子查询中查找

- **如果在子查询中存在满足条件的行：**
  - 不在子查询中继续查找
  - 条件返回 TRUE

- NOT EXISTS关键字表示如果不存在某种条件，则返回TRUE，否则返回FALSE

题目：查询公司管理者的employee_id，last_name，job_id，department_id信息

方式一：相关子查询

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees e1
WHERE  EXISTS ( SELECT *
                FROM   employees e2
                WHERE  e2.manager_id = e1.employee_id
              );

+-------------+-----------+---------+---------------+
| employee_id | last_name | job_id  | department_id |
+-------------+-----------+---------+---------------+
|         100 | King      | AD_PRES |            90 |
|         101 | Kochhar   | AD_VP   |            90 |
|         102 | De Haan   | AD_VP   |            90 |
|         103 | Hunold    | IT_PROG |            60 |
|         201 | Hartstein | MK_MAN  |            20 |
|         205 | Higgins   | AC_MGR  |           110 |
+-------------+-----------+---------+---------------+
18 rows in set (0.00 sec)
```

方式二：自连接

同一个表形成两个虚拟表；一个用于获取员工信息；一个获取管理者id（确定哪个员工是管理者）

```
SELECT DISTINCT e1.employee_id, e1.last_name, e1.job_id, e1.department_id
FROM   employees e1 JOIN employees e2
WHERE e1.employee_id = e2.manager_id;

+-------------+-----------+---------+---------------+
| employee_id | last_name | job_id  | department_id |
+-------------+-----------+---------+---------------+
|         100 | King      | AD_PRES |            90 |
|         101 | Kochhar   | AD_VP   |            90 |
|         149 | Zlotkey   | SA_MAN  |            80 |
|         201 | Hartstein | MK_MAN  |            20 |
|         205 | Higgins   | AC_MGR  |           110 |
+-------------+-----------+---------+---------------+
18 rows in set (0.00 sec)
```

方式三：EXISTS

```
SELECT employee_id,last_name,job_id,department_id
FROM employees
WHERE employee_id IN (
             SELECT DISTINCT manager_id
             FROM employees
             );
```

题目：查询departments表中，不存在于employees表中的部门的department_id和department_name

题目解读：就是想找哪些部门没有人

方式一：子查询 NOT EXISTS

```
SELECT department_id, department_name
FROM departments d
WHERE NOT EXISTS (SELECT 'X'
                  FROM   employees
                  WHERE  department_id = d.department_id);

+---------------+----------------------+
| department_id | department_name      |
+---------------+----------------------+
|           120 | Treasury             |
|           130 | Corporate Tax        |
|           240 | Government Sales     |
|           250 | Retail Sales         |
|           260 | Recruiting           |
|           270 | Payroll              |
+---------------+----------------------+
16 rows in set (0.00 sec)
```

方式二：左中图或者是右中图

```
SELECT d.department_id,d.department_name
FROM employees e RIGHT JOIN departments d
ON e.department_id=d.department_id
WHERE e.department_id IS NULL;

+---------------+----------------------+
| department_id | department_name      |
+---------------+----------------------+
|           120 | Treasury             |
|           130 | Corporate Tax        |
|           140 | Control And Credit   |
|           240 | Government Sales     |
|           250 | Retail Sales         |
```

```
|           260 | Recruiting           |
|           270 | Payroll              |
+---------------+----------------------+
16 rows in set (0.00 sec)
```

4.4 相关更新

```
UPDATE table1 alias1
SET    column = (SELECT expression
                 FROM   table2 alias2
                 WHERE  alias1.column = alias2.column);
```

使用相关子查询依据一个表中的数据更新另一个表的数据。

题目：在employees中增加一个department_name字段，数据为员工对应的部门名称

```
# 1)
ALTER TABLE employees
ADD(department_name VARCHAR2(14));

# 2)
UPDATE employees e
SET department_name =  (SELECT department_name
                        FROM   departments d
                        WHERE  e.department_id = d.department_id);
```

4.4 相关删除

```
 DELETE FROM table1 alias1
 WHERE column operator (SELECT expression
                        FROM   table2 alias2
                        WHERE  alias1.column = alias2.column);
```

使用相关子查询依据一个表中的数据删除另一个表的数据。

题目：删除表employees中，其与emp_history表皆有的数据

```
DELETE FROM employees e
WHERE employee_id in
           (SELECT employee_id
            FROM   emp_history
            WHERE  employee_id = e.employee_id);
```

5. 抛一个思考题

问题：谁的工资比Abel的高?

解答：

```
#方式1：自连接
SELECT e2.last_name,e2.salary
FROM employees e1,employees e2
WHERE e1.last_name = 'Abel'
AND e1.`salary` < e2.`salary`
```

```
#方式2：子查询
SELECT last_name,salary
FROM employees
WHERE salary > (
        SELECT salary
        FROM employees
        WHERE last_name = 'Abel'
        );
```

问题：以上两种方式有好坏之分吗?

解答：自连接方式好!

题目中可以使用子查询，也可以使用自连接。一般情况建议你使用自连接，因为在许多 DBMS 的处理过程中，对于自连接的处理速度要比子查询快得多

可以这样理解：子查询实际上是通过未知表进行查询后的条件判断，而自连接是通过已知的自身数据表进行条件判断，因此在大部分 DBMS 中都对自连接处理进行了优化

课后练习

1.查询和Zlotkey相同部门的员工姓名和工资

不相关子查询

```
SELECT last_name, salary
FROM employees
WHERE department_id = (
                    SELECT department_id
                    FROM employees
                    WHERE last_name = 'Zlotkey'
                    );

+------------+----------+
```

```
| last_name  | salary   |
+------------+----------+
| Russell    | 14000.00 |
| Partners   | 13500.00 |
| Errazuriz  | 12000.00 |
| Taylor     |  8600.00 |
| Livingston |  8400.00 |
| Johnson    |  6200.00 |
+------------+----------+
34 rows in set (0.00 sec)
```

自连接：

两张表，一张表用于确定员工信息，一张表用于确定Zlotkey的相关信息

```
SELECT e1.last_name, e1.salary
FROM employees e1 INNER JOIN employees e2
WHERE e2.last_name='Zlotkey'
AND e1.department_id=e2.department_id;

+------------+----------+
| last_name  | salary   |
+------------+----------+
| Russell    | 14000.00 |
| Partners   | 13500.00 |
| Livingston |  8400.00 |
| Johnson    |  6200.00 |
+------------+----------+
34 rows in set (0.00 sec)
```

2.查询工资比公司平均工资高的员工的员工号，姓名和工资

```
SELECT employee_id, last_name, salary
FROM employees
WHERE salary > (
                SELECT AVG(salary)
                FROM employees
                );

+-------------+------------+----------+
| employee_id | last_name  | salary   |
+-------------+------------+----------+
|         100 | King       | 24000.00 |
|         204 | Baer       | 10000.00 |
|         205 | Higgins    | 12000.00 |
|         206 | Gietz      |  8300.00 |
```

```
+------------+-----------+----------+
51 rows in set (0.00 sec)
```

3.选择工资大于所有JOB_ID = 'SA_MAN'的员工的工资的员工的last_name, job_id, salary

```
SELECT last_name,job_id,salary
FROM employees
-- 所有
WHERE salary > ALL (
                    SELECT salary
                    FROM employees
                    WHERE job_id = 'SA_MAN'
                    );

+-----------+---------+----------+
| last_name | job_id  | salary   |
+-----------+---------+----------+
| King      | AD_PRES | 24000.00 |
| Kochhar   | AD_VP   | 17000.00 |
| De Haan   | AD_VP   | 17000.00 |
+-----------+---------+----------+
3 rows in set (0.00 sec)
```

4.查询姓名中包含字母u的员工在相同部门的员工的员工号和姓名

查询 名字中有u的人所在的部门中员工的信息

```
SELECT employee_id, last_name
FROM employees
WHERE department_id = ANY(
                        SELECT DISTINCT department_id
                        FROM employees
                        WHERE last_name LIKE '%u%'
                        );

+-------------+-------------+
| employee_id | last_name   |
+-------------+-------------+
|         103 | Hunold      |
|         104 | Ernst       |
|         176 | Taylor      |
|         177 | Livingston  |
|         179 | Johnson     |
+-------------+-------------+
96 rows in set (0.00 sec)
```

5.查询在部门的location_id为1700的部门工作的员工的员工号

```
SELECT employee_id
FROM employees
WHERE department_id IN (
                    SELECT department_id
                    FROM departments
                    WHERE location_id = 1700
                    );

+-------------+
| employee_id |
+-------------+
|         200 |
|         206 |
+-------------+
18 rows in set (0.00 sec)
```

6.查询管理者是King的员工姓名和工资

方式一：

```
SELECT last_name, salary
FROM employees
WHERE manager_id IN (
                    SELECT employee_id
                    FROM employees
                    WHERE last_name = 'King'
                    );

+-----------+----------+
| last_name | salary   |
+-----------+----------+
| Kochhar   | 17000.00 |
| De Haan   | 17000.00 |
| Zlotkey   | 10500.00 |
| Hartstein | 13000.00 |
+-----------+----------+
14 rows in set (0.00 sec)
```

方式二：子查询EXISSTS

```
SELECT last_name, salary
FROM employees e1
```

```
WHERE EXISTS (
                    SELECT 'X'
                    FROM dual
                    WHERE e1.manager_id IN
                    ( SELECT employee_id
                    FROM employees
                    WHERE last_name = 'King')
                    );

+-----------+----------+
| last_name | salary   |
+-----------+----------+
| Kochhar   | 17000.00 |
| De Haan   | 17000.00 |
| Raphaely  | 11000.00 |
| Cambrault | 11000.00 |
| Zlotkey   | 10500.00 |
| Hartstein | 13000.00 |
+-----------+----------+
14 rows in set (0.00 sec)
```

7. 查询工资最低的员工信息: last_name, salary

```
SELECT last_name,salary
FROM employees
WHERE salary = (
                SELECT MIN(salary)
                FROM employees
                );

+-----------+---------+
| last_name | salary  |
+-----------+---------+
| Olson     | 2100.00 |
+-----------+---------+
1 row in set (0.00 sec)
```

8. 查询平均工资最低的部门信息

方式一:

```
-- 各个部门的平均工资
SELECT department_id,AVG(salary) avg_salary
FROM employees
WHERE department_id IS NOT NULL
GROUP BY department_id;
```

```
-- 获取最低工资的部门的id
SELECT t.department_id ,MIN(avg_salary)
FROM (
    SELECT department_id,AVG(salary) avg_salary
    FROM employees
    WHERE department_id IS NOT NULL
    GROUP BY department_id
      ) t

+---------------+-----------------+
| department_id | MIN(avg_salary) |
+---------------+-----------------+
|            10 |     3475.555556 |
+---------------+-----------------+
1 row in set (0.00 sec)



-- 多表查询
SELECT * FROM departments d,(
                SELECT t.department_id ,MIN(avg_salary)
                FROM (
                    SELECT department_id,AVG(salary) avg_salary
                    FROM employees
                    WHERE department_id IS NOT NULL
                    GROUP BY department_id
                      ) t

)tt
 WHERE d.department_id=tt.department_id;

+---------------+-----------------+------------+-------------+---------------
+-----------------+
| department_id | department_name | manager_id | location_id | department_id |
MIN(avg_salary) |
+---------------+-----------------+------------+-------------+---------------
+-----------------+
|            10 | Administration  |        200 |        1700 |            10 |
    3475.555556 |
+---------------+-----------------+------------+-------------+---------------
+-----------------+
1 row in set (0.00 sec)
```

方式二：

```
-- 各部门平均工资
SELECT AVG(salary) dept_avgsal
FROM employees
```

```
GROUP BY department_id;

+--------------+
| dept_avgsal  |
+--------------+
|  7000.000000 |
| 10150.000000 |
+--------------+
12 rows in set (0.00 sec)


-- 通过HAVING过滤 获得平均工资最小的部门id
SELECT department_id
FROM employees
GROUP BY department_id
HAVING AVG(salary)<=ALL(
                        SELECT AVG(salary) dept_avgsal
                        FROM employees
                        GROUP BY department_id
                        );
# 注意：是小于等于 没有等于空集
+---------------+
| department_id |
+---------------+
|            50 |
+---------------+
1 row in set (0.00 sec)



-- 根据部门id获取信息
SELECT * FROM departments
WHERE department_id=(
                    SELECT department_id
                    FROM employees
                    GROUP BY department_id
                    HAVING AVG(salary)<=ALL(
                                            SELECT AVG(salary) dept_avgsal
                                            FROM employees
                                            GROUP BY department_id
                                            )
);

+---------------+-----------------+------------+-------------+
| department_id | department_name | manager_id | location_id |
+---------------+-----------------+------------+-------------+
|            50 | Shipping        |        121 |        1500 |
+---------------+-----------------+------------+-------------+
```

方式三：

```
-- 各部门平均工资
SELECT AVG(salary) dept_avgsal
FROM employees
GROUP BY department_id;

+--------------+
| dept_avgsal  |
+--------------+
|  7000.000000 |
| 10150.000000 |
+--------------+
12 rows in set (0.00 sec)


-- 获取各部门中最小的平均工资
 SELECT MIN(dept_avgsal)
FROM (
        SELECT AVG(salary) dept_avgsal
        FROM employees
        GROUP BY department_id
        ) avg_sal;

+------------------+
| MIN(dept_avgsal) |
+------------------+
|      3475.555556 |
+------------------+

-- 获取各部门中最小的部门id
SELECT department_id
FROM employees
GROUP BY department_id
HAVING AVG(salary) = (
                        SELECT MIN(dept_avgsal)
                                    FROM (
                                    SELECT AVG(salary) dept_avgsal
                                    FROM employees
                                    GROUP BY department_id
                                    ) avg_sal
                        );

+---------------+
| department_id |
+---------------+
|            50 |
+---------------+
1 row in set (0.00 sec)

SELECT *
FROM departments
WHERE department_id = (
                          SELECT department_id
                          FROM employees
                          GROUP BY department_id
```

```
                        HAVING AVG(salary) = (
                                                SELECT MIN(dept_avgsal)
                                                        FROM (
                                                        SELECT
AVG(salary) dept_avgsal
                                                        FROM employees
                                                        GROUP BY
department_id
                                                        ) avg_sal
                                                )
                        );

+---------------+-----------------+------------+-------------+
| department_id | department_name | manager_id | location_id |
+---------------+-----------------+------------+-------------+
|            50 | Shipping        |        121 |        1500 |
+---------------+-----------------+------------+-------------+
1 row in set (0.00 sec)
```

方式四：用LIMIT获取最小值

```
-- 各部门中，最小的平均工资
SELECT department_id,AVG(salary) dept_avgsal
FROM employees
GROUP BY department_id
ORDER BY dept_avgsal
LIMIT 1;

+---------------+-------------+
| department_id | dept_avgsal |
+---------------+-------------+
|            50 | 3475.555556 |
+---------------+-------------+
1 row in set (0.00 sec)


-- 多表查询
SELECT d.*
FROM departments d INNER JOIN (
                                SELECT department_id,AVG(salary) dept_avgsal
                                FROM employees
                                GROUP BY department_id
                                ORDER BY dept_avgsal
                                LIMIT 1
                                ) t_d
WHERE d.department_id=t_d.department_id;

+---------------+-----------------+------------+-------------+
| department_id | department_name | manager_id | location_id |
+---------------+-----------------+------------+-------------+
```

```
|            50 | Shipping        |        121 |        1500 |
+---------------+-----------------+------------+-------------+
1 row in set (0.00 sec)
```

9.查询平均工资最低的部门信息和该部门的平均工资（相关子查询）

```
-- 各部门中，最小的平均工资
SELECT department_id,AVG(salary) dept_avgsal
FROM employees
GROUP BY department_id
ORDER BY dept_avgsal
LIMIT 1;

+---------------+-------------+
| department_id | dept_avgsal |
+---------------+-------------+
|            50 | 3475.555556 |
+---------------+-------------+
1 row in set (0.00 sec)


-- 多表查询
SELECT d.*,t_d. dept_avgsal
FROM departments d INNER JOIN (
                               SELECT department_id,AVG(salary) dept_avgsal
                               FROM employees
                               GROUP BY department_id
                               ORDER BY dept_avgsal
                               LIMIT 1
                               ) t_d
WHERE d.department_id=t_d.department_id;

+---------------+-----------------+------------+-------------+-------------+
| department_id | department_name | manager_id | location_id | dept_avgsal |
+---------------+-----------------+------------+-------------+-------------+
|            50 | Shipping        |        121 |        1500 | 3475.555556 |
+---------------+-----------------+------------+-------------+-------------+
1 row in set (0.00 sec)
```

```
-- 方式二：
SELECT d.*,(SELECT AVG(salary) FROM employees WHERE department_id =
d.department_id)
avg_sal
FROM departments d
WHERE department_id = (
                       SELECT department_id
                       FROM employees
```

```
                        GROUP BY department_id
                        HAVING AVG(salary) = (
                                                SELECT AVG(salary) avg_sal
                                                FROM employees
                                                GROUP BY department_id
                                                ORDER BY avg_sal
                                                LIMIT 0,1
                                                )
                        );
```

```
+---------------+-----------------+------------+-------------+-------------+
| department_id | department_name | manager_id | location_id | avg_sal     |
+---------------+-----------------+------------+-------------+-------------+
|            50 | Shipping        |        121 |        1500 | 3475.555556 |
+---------------+-----------------+------------+-------------+-------------+
1 row in set (0.00 sec)
```

10. 查询平均工资最高的 job 信息

```
SELECT * FROM jobs;
```

```
+------------+---------------------------------+------------+------------+
| job_id     | job_title                       | min_salary | max_salary |
+------------+---------------------------------+------------+------------+
| AC_ACCOUNT | Public Accountant               |       4200 |       9000 |
| AC_MGR     | Accounting Manager              |       8200 |      16000 |
| AD_ASST    | Administration Assistant        |       3000 |       6000 |
| AD_PRES    | President                       |      20000 |      40000 |
| AD_VP      | Administration Vice President   |      15000 |      30000 |
| FI_ACCOUNT | Accountant                      |       4200 |       9000 |
| FI_MGR     | Finance Manager                 |       8200 |      16000 |
| HR_REP     | Human Resources Representative  |       4000 |       9000 |
| IT_PROG    | Programmer                      |       4000 |      10000 |
| MK_MAN     | Marketing Manager               |       9000 |      15000 |
| MK_REP     | Marketing Representative        |       4000 |       9000 |
| PR_REP     | Public Relations Representative |       4500 |      10500 |
| PU_CLERK   | Purchasing Clerk                |       2500 |       5500 |
| PU_MAN     | Purchasing Manager              |       8000 |      15000 |
| SA_MAN     | Sales Manager                   |      10000 |      20000 |
| SA_REP     | Sales Representative            |       6000 |      12000 |
| SH_CLERK   | Shipping Clerk                  |       2500 |       5500 |
| ST_CLERK   | Stock Clerk                     |       2000 |       5000 |
| ST_MAN     | Stock Manager                   |       5500 |       8500 |
+------------+---------------------------------+------------+------------+
19 rows in set (0.00 sec)
```

方式一：

```
SELECT *
```

```
FROM jobs
WHERE job_id = (
                    SELECT job_id
                    FROM employees
                    GROUP BY job_id
                    HAVING AVG(salary) = (
                                            SELECT MAX(avg_sal)
                                            FROM(
                                                    SELECT AVG(salary) avg_sal
                                                    FROM employees
                                                    GROUP BY job_id
                                                    ) job_avgsal #结果集作为表出
现记得起别名
                                            )
                    );

+---------+-----------+------------+------------+
| job_id  | job_title | min_salary | max_salary |
+---------+-----------+------------+------------+
| AD_PRES | President |      20000 |      40000 |
+---------+-----------+------------+------------+
1 row in set (0.01 sec)
```

方式二：

```
-- 获取平均工资最高的job_id
SELECT job_id ,AVG(salary) avg_salary
FROM employees
GROUP BY job_id
ORDER BY avg_salary DESC
LIMIT 1;

+---------+--------------+
| job_id  | avg_salary   |
+---------+--------------+
| AD_PRES | 24000.000000 |
+---------+--------------+
1 row in set (0.00 sec)

-- 多表查询

SELECT j.*
FROM jobs j,
            (
            SELECT job_id ,AVG(salary) avg_salary
            FROM employees
            GROUP BY job_id
            ORDER BY avg_salary DESC
            LIMIT 1
            ) t_j
WHERE j.job_id=t_j.job_id;
```

```
+---------+-----------+------------+------------+
| job_id  | job_title | min_salary | max_salary |
+---------+-----------+------------+------------+
| AD_PRES | President |      20000 |      40000 |
+---------+-----------+------------+------------+
1 row in set (0.00 sec)
```

11. 查询平均工资高于公司平均工资的部门有哪些?

```
-- 公司的平均工资
SELECT AVG(salary)
FROM employees ;

+-------------+
| AVG(salary) |
+-------------+
| 6461.682243 |
+-------------+
1 row in set (0.00 sec)


--  查询平均工资高于公司平均工资的部门

SELECT department_id
FROM employees
WHERE department_id IS NOT NULL
GROUP BY department_id
HAVING AVG(salary)>(
                    SELECT AVG(salary)
                    FROM employees
                    );
+---------------+
| department_id |
+---------------+
|            20 |
|            40 |
|            70 |
|            80 |
|            90 |
|           100 |
|           110 |
+---------------+
7 rows in set (0.00 sec)
```

12. 查询出公司中所有 manager 的详细信息

方式一:

```
SELECT last_name ,salary,job_id
```

```
FROM employees
WHERE employee_id IN (
    SELECT DISTINCT manager_id
    FROM employees
    WHERE manager_id IS NOT NULL
);

+-----------+----------+---------+
| last_name | salary   | job_id  |
+-----------+----------+---------+
| King      | 24000.00 | AD_PRES |
| Kochhar   | 17000.00 | AD_VP   |
| De Haan   | 17000.00 | AD_VP   |
| Hunold    |  9000.00 | IT_PROG |
| Greenberg | 12000.00 | FI_MGR  |
| Raphaely  | 11000.00 | PU_MAN  |
| Weiss     |  8000.00 | ST_MAN  |
| Fripp     |  8200.00 | ST_MAN  |
| Kaufling  |  7900.00 | ST_MAN  |
| Vollman   |  6500.00 | ST_MAN  |
| Mourgos   |  5800.00 | ST_MAN  |
| Russell   | 14000.00 | SA_MAN  |
| Partners  | 13500.00 | SA_MAN  |
| Errazuriz | 12000.00 | SA_MAN  |
| Cambrault | 11000.00 | SA_MAN  |
| Zlotkey   | 10500.00 | SA_MAN  |
| Hartstein | 13000.00 | MK_MAN  |
| Higgins   | 12000.00 | AC_MGR  |
+-----------+----------+---------+
18 rows in set (0.00 sec)
```

注意：一个小结论，写IN的地方一般可以改写为EXISTS

方式二：EXISTS

```
SELECT last_name ,salary,job_id
FROM employees e1
WHERE  EXISTS (
    SELECT '1'
    FROM employees e2
    WHERE e1.employee_id=e2.manager_id #对比当前记录的manager_id，与外面传进来的是否相等  相当于两层for
);

+-----------+----------+---------+
| last_name | salary   | job_id  |
+-----------+----------+---------+
| King      | 24000.00 | AD_PRES |
| Kochhar   | 17000.00 | AD_VP   |
| De Haan   | 17000.00 | AD_VP   |
| Hunold    |  9000.00 | IT_PROG |
| Greenberg | 12000.00 | FI_MGR  |
```

```
| Raphaely  | 11000.00 | PU_MAN  |
| Weiss     |  8000.00 | ST_MAN  |
| Fripp     |  8200.00 | ST_MAN  |
| Kaufling  |  7900.00 | ST_MAN  |
| Vollman   |  6500.00 | ST_MAN  |
| Mourgos   |  5800.00 | ST_MAN  |
| Russell   | 14000.00 | SA_MAN  |
| Partners  | 13500.00 | SA_MAN  |
| Errazuriz | 12000.00 | SA_MAN  |
| Cambrault | 11000.00 | SA_MAN  |
| Zlotkey   | 10500.00 | SA_MAN  |
| Hartstein | 13000.00 | MK_MAN  |
| Higgins   | 12000.00 | AC_MGR  |
+-----------+----------+---------+
18 rows in set (0.00 sec)
```

方式三：自连接

13. 各个部门中 最高工资中最低的那个部门的 最低工资是多少?

解析：拿各个部门的最高工资进行比较，得到最低工资的的部门，再获取该部门的最低工资

```
-- 最低工资的部门id
SELECT department_id ,MAX(salary) max_salary
FROM employees
GROUP BY department_id
ORDER BY max_salary
LIMIT 1;

+---------------+------------+
| department_id | max_salary |
+---------------+------------+
|            10 |    4400.00 |
+---------------+------------+
1 row in set (0.00 sec)



SELECT e.department_id,MIN(salary)
FROM employees e,
                (
                SELECT department_id ,MAX(salary) max_salary
                FROM employees
                GROUP BY department_id
                ORDER BY max_salary
                LIMIT 1
                ) t_e
WHERE e.department_id=t_e.department_id;
```

```
+---------------+-------------+
| department_id | MIN(salary) |
+---------------+-------------+
|            10 |     4400.00 |
+---------------+-------------+
1 row in set (0.00 sec)
```

14. 查询平均工资最高的部门的 manager 的详细信息: last_name, department_id, email, salary

```
-- 平均工资最高的部门id
SELECT department_id ,MAX(salary) max_salary
FROM employees
GROUP BY department_id
ORDER BY max_salary DESC
LIMIT 1;

+---------------+------------+
| department_id | max_salary |
+---------------+------------+
|            90 |   24000.00 |
+---------------+------------+
1 row in set (0.00 sec)

--  查询平均工资最高的部门的 manager_id
SELECT DISTINCT e.manager_id
FROM
employees e,(
                        SELECT department_id ,MAX(salary) max_salary
                        FROM employees
                        GROUP BY department_id
                        ORDER BY max_salary DESC
                        LIMIT 1
                        )t_e,
                        employees e2
WHERE e.department_id=t_e.department_id
AND e.manager_id IS NOT NULL;

+------------+
| manager_id |
+------------+
|        100 |
+------------+
1 row in set (0.00 sec)


-- 根据employee的id获取详细信息
```

```
SELECT last_name, department_id, email, salary
FROM employees
WHERE manager_id=(
                    SELECT DISTINCT e.manager_id
                    FROM
                    employees e,(
                                            SELECT department_id ,MAX(salary)
max_salary
                                            FROM employees
                                            GROUP BY department_id
                                            ORDER BY max_salary DESC
                                            LIMIT 1
                                            )t_e,
                                            employees e2
                    WHERE e.department_id=t_e.department_id
                    AND e.manager_id IS NOT NULL
                    );
```

```
+-----------+---------------+----------+----------+
| last_name | department_id | email    | salary   |
+-----------+---------------+----------+----------+
| Kochhar   |            90 | NKOCHHAR | 17000.00 |
| De Haan   |            90 | LDEHAAN  | 17000.00 |
| Raphaely  |            30 | DRAPHEAL | 11000.00 |
| Weiss     |            50 | MWEISS   |  8000.00 |
| Fripp     |            50 | AFRIPP   |  8200.00 |
| Kaufling  |            50 | PKAUFLIN |  7900.00 |
| Vollman   |            50 | SVOLLMAN |  6500.00 |
| Mourgos   |            50 | KMOURGOS |  5800.00 |
| Russell   |            80 | JRUSSEL  | 14000.00 |
| Partners  |            80 | KPARTNER | 13500.00 |
| Errazuriz |            80 | AERRAZUR | 12000.00 |
| Cambrault |            80 | GCAMBRAU | 11000.00 |
| Zlotkey   |            80 | EZLOTKEY | 10500.00 |
| Hartstein |            20 | MHARTSTE | 13000.00 |
+-----------+---------------+----------+----------+
14 rows in set (0.00 sec)
```

15. 查询部门的部门号，其中不包括job_id是"ST_CLERK"的部门号

方式一：NOT EXISTS

```
SELECT department_id
FROM departments d
WHERE NOT EXISTS(
SELECT '1'
    FROM
    employees e
    WHERE e.department_id=d.department_id
    AND e.job_id='ST_CLERK'
```

```
);

+---------------+
| department_id |
+---------------+
|            10 |
|            20 |
|           250 |
|           260 |
|           270 |
+---------------+
26 rows in set (0.00 sec)
```

方式二：NOT IN

```
SELECT department_id
FROM departments d
WHERE department_id NOT IN(
SELECT department_id
    FROM
    employees
    WHERE job_id='ST_CLERK'
);

+---------------+
| department_id |
+---------------+
|            10 |
|            20 |
|           260 |
|           270 |
+---------------+
26 rows in set (0.00 sec)
```

方式三：单表查询

16. 选择所有没有管理者的员工的last_name

正经方式：

```
SELECT last_name
FROM employees
WHERE manager_id IS  NULL;

+-----------+
| last_name |
+-----------+
| King      |
+-----------+
1 row in set (0.00 sec)
```

NOT EXISTS

```
SELECT last_name
FROM employees e1
WHERE NOT EXISTS(
                    SELECT '1'
                    FROM employees e2
                    WHERE e1.manager_id=e2.employee_id -- 外表的管理者信息是否存在
于员工表中
                );

+-----------+
| last_name |
+-----------+
| King      |
+-----------+
1 row in set (0.00 sec)
```

17．查询员工号、姓名、雇用时间、工资，其中员工的管理者为 'De Haan'

方式1：

```
SELECT employee_id, last_name, hire_date, salary
FROM employees
WHERE manager_id IN (
SELECT employee_id
FROM employees
WHERE last_name = 'De Haan'
);

+-------------+-----------+-----------+---------+
| employee_id | last_name | hire_date  | salary  |
+-------------+-----------+-----------+---------+
|         103 | Hunold    | 1990-01-03 | 9000.00 |
```

```
+-------------+-----------+-----------+---------+
1 row in set (0.00 sec)
```

方式2：

```
SELECT employee_id, last_name, hire_date, salary
FROM employees e1
WHERE EXISTS (
SELECT *
FROM employees e2
WHERE e2.`employee_id` = e1.manager_id
AND e2.last_name = 'De Haan'
);

+-------------+-----------+-----------+---------+
| employee_id | last_name | hire_date  | salary  |
+-------------+-----------+-----------+---------+
|         103 | Hunold    | 1990-01-03 | 9000.00 |
+-------------+-----------+-----------+---------+
1 row in set (0.00 sec)
```

18.查询各部门中工资比本部门平均工资高的员工的员工号，姓名和工资

方式一：相关子查询

```
SELECT employee_id,last_name,salary
FROM employees e1
WHERE salary > (
                # 查询某员工所在部门的平均
                SELECT AVG(salary)
                FROM employees e2
                WHERE e2.department_id = e1.`department_id`
                );
```

方式二：

```
SELECT employee_id,last_name,salary
FROM employees e1,
(SELECT department_id,AVG(salary) avg_sal
FROM employees e2 GROUP BY department_id
) dept_avg_sal
WHERE e1.`department_id` = dept_avg_sal.department_id
AND e1.`salary` > dept_avg_sal.avg_sal;
```

19.查询每个部门下的部门人数大于 5 的部门名称

相关子查询

```
SELECT department_name,department_id
FROM departments d
WHERE 5 < (
            SELECT COUNT(*)
            FROM employees e
            WHERE d.`department_id` = e.`department_id`
            );

+-----------------+---------------+
| department_name | department_id |
+-----------------+---------------+
| Purchasing      |            30 |
| Shipping        |            50 |
| Sales           |            80 |
| Finance         |           100 |
+-----------------+---------------+
4 rows in set (0.00 sec)
```

20.查询每个国家下的部门个数大于 2 的国家编号

```
SELECT * FROM locations;

+-------------+----------------------------------------+-------------+------
---------------+-------------------+------------+
| location_id | street_address                         | postal_code | city
               | state_province    | country_id |
+-------------+----------------------------------------+-------------+------
---------------+-------------------+------------+
|        1000 | 1297 Via Cola di Rie                   | 00989       | Roma
               | NULL              | IT         |
|        1100 | 93091 Calle della Testa                | 10934       |
Venice              | NULL              | IT         |
|        1200 | 2017 Shinjuku-ku                       | 1689        | Tokyo
              | Tokyo Prefecture  | JP         |
|        1300 | 9450 Kamiya-cho                        | 6823        |
Hiroshima           | NULL              | JP         |
+-------------+----------------------------------------+-------------+------
---------------+-------------------+------------+
23 rows in set (0.01 sec)
```

相关子查询

```
SELECT country_id
FROM locations l
WHERE 2 < (
            SELECT COUNT(*)
            FROM departments d
            WHERE l.`location_id` = d.`location_id`
            );
```

子查询编写技巧：

1、从里往外写：当子查询较难

2、从外往里写：当是相关子查询、当子查询较简单