中国石油大学 **(北京)**

CHINA UNIVERSITY OF PETROLEUM

数据库原理上机报告

实 T-SQL 编程验三 **T-SQL 编程**

院系名称：**信息学院**

专业名称：**计算机**

学生学号：2019011777

学生姓名：刘康来

完成日期 2021 年 11 月 17 日

《数据库原理》上机报告

报告名称	*T-SQL 编程*			**粘贴照片**
姓 名	刘康来	**班 级**	计算机 19-3	
成 绩				

1．上机实践目的与准备知识（简介，300 字以内）

1.实验目的

（1）掌握变量、基本数据类型、运算符、控制语句的基本语法和使用；
（2）掌握系统函数和用户自定义函数的使用；
（3）掌握存储过程的基本格式和使用。

2. 实验准备

（1）了解 T-SQL 支持的变量、基本数据类型、运算符、控制语句的基本语法和使用方法；
（2）了解系统函数的调用方法和用户自定义函数的使用步骤；
（3）了解存储过程的编写方法和使用过程。

2．主要实践内容与具体操作步骤（实践内容完成情况要有描述，如执行的 SQL 命令等，有运行结果截图，图大小以保证文字清晰为准）

（1）自定义数据类型

```
CREATE TYPE Employee_num
 FROM char(6) NOT NULL
```

练习 1：使用 SQL 语句创建表 Employees3,结构与表 Employees 类似，只是 EmployeeID 使用上述自定义类型 Employee_num；

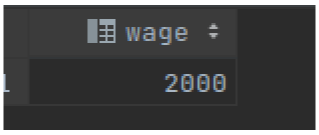
mysql 没有。。。

（2）使用变量

例如：DECLARE @student char(6)
例如：SELECT @var1 =
(
 SELECT 姓名
 FROM xsb
 WHERE 学号= '191399'
)
SELECT @var1 AS 'NAME'

练习 2： 定义一个变量，描述 Salary 表某一员工（员工号根据表中实际值选定）的实际收入（income-outcome），然后查询该变量。

```
set @wage = (
    select Income - Outcome
    from Salary
    where EmployeeID = '1'
);

select @wage as wage;
```



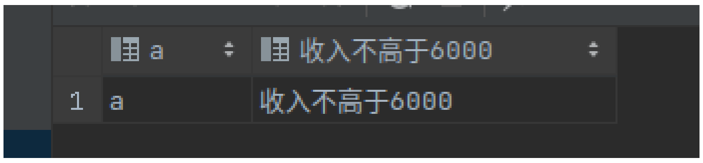
```
  wage
1 2000
```

（3）流程控制

分支结构练习 3.1： 判断姓名为“王琳” （或其他名字）的员工实际收入是否高于 6000，如果是则显示其收入，否则显示“收入不高于 6000”。

```
create procedure judge()
begin
    if @wagea > 6000
        then  select 'a', @wagea;
    else
         select 'a', '收入不高于6000';
    end if;
end;
```

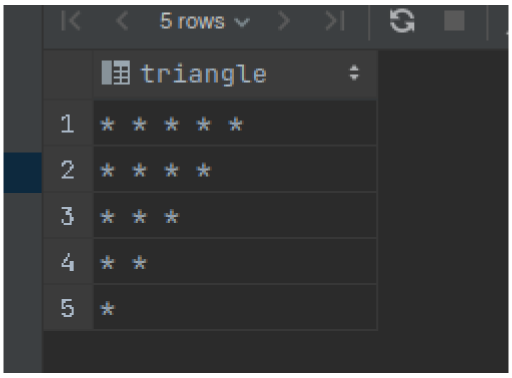
```
call judge();
```



	a	收入不高于6000
1	a	收入不高于6000

循环结构练习 3.2：
（a）使用循环输出一个用’ *’组成的 5 行三角形。

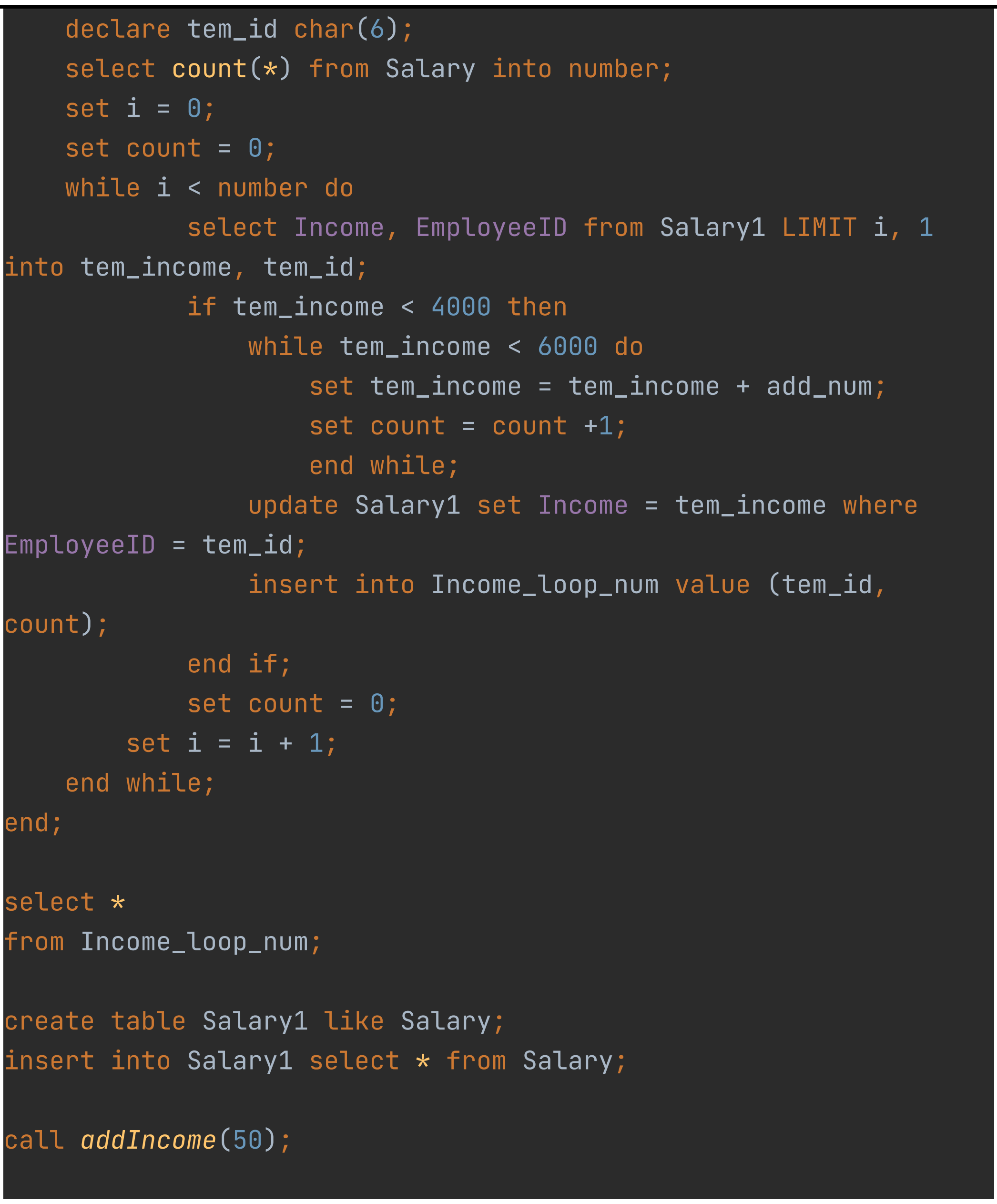
```
-- mysql haven't print!!!

SET @NUMBER = 6;
SELECT REPEAT('* ', @NUMBER := @NUMBER - 1) as 'triangle'
    FROM information_schema.tables LIMIT 5;
```



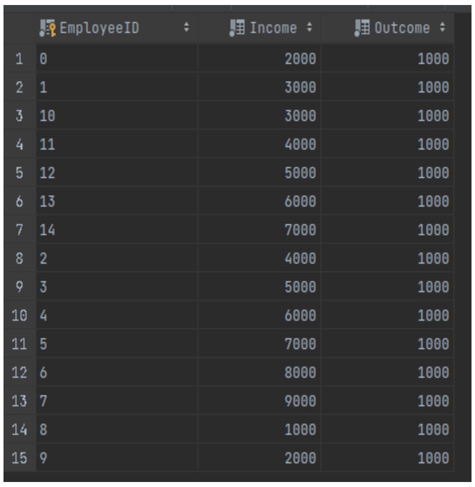
	triangle
1	* * * * *
2	* * * *
3	* * *
4	* *
5	*

（b）将员工收入低于 4000 的员工的收入使用循环修改到 6000，每次只加 50，并判断循环了多少次。

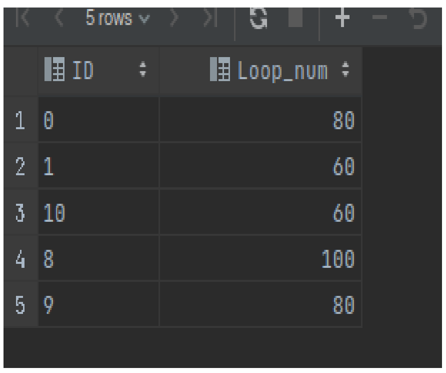
```
create table Income_loop_num(
    ID char(6),
    Loop_num int
);

create procedure addIncome(add_num int)
begin
    declare number int;
    declare i int;
    declare count int;
    declare tem_income int;
```



```
    declare tem_id char(6);
    select count(*) from Salary into number;
    set i = 0;
    set count = 0;
    while i < number do
            select Income, EmployeeID from Salary1 LIMIT i, 1
into tem_income, tem_id;
            if tem_income < 4000 then
                while tem_income < 6000 do
                    set tem_income = tem_income + add_num;
                    set count = count +1;
                    end while;
                update Salary1 set Income = tem_income where
EmployeeID = tem_id;
                insert into Income_loop_num value (tem_id,
count);
            end if;
            set count = 0;
        set i = i + 1;
    end while;
end;

select *
from Income_loop_num;

create table Salary1 like Salary;
insert into Salary1 select * from Salary;

call addIncome(50);
```



	EmployeeID	Income	Outcome
1	0	2000	1000
2	1	3000	1000
3	10	3000	1000
4	11	4000	1000
5	12	5000	1000
6	13	6000	1000
7	14	7000	1000
8	2	4000	1000
9	3	5000	1000
10	4	6000	1000
11	5	7000	1000
12	6	8000	1000
13	7	9000	1000
14	8	1000	1000
15	9	2000	1000

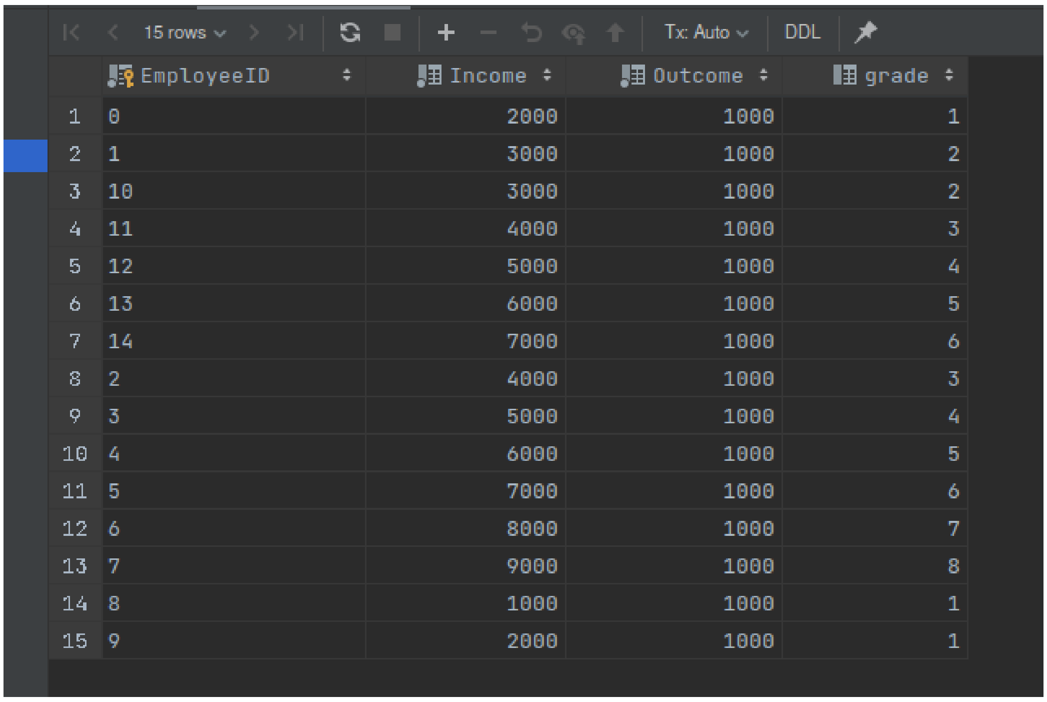


	ID	Loop_num
1	0	80
2	1	60
3	10	60
4	8	100
5	9	80

（4）自定义函数

练习 4：a）编写一个函数用来对员工的工资进行分级，3000 元以下为 1 级，3000-4000 元为 2 级，…以此类推，每级相差 1000 元。调用该函数显示每个员工的工资及其级别。

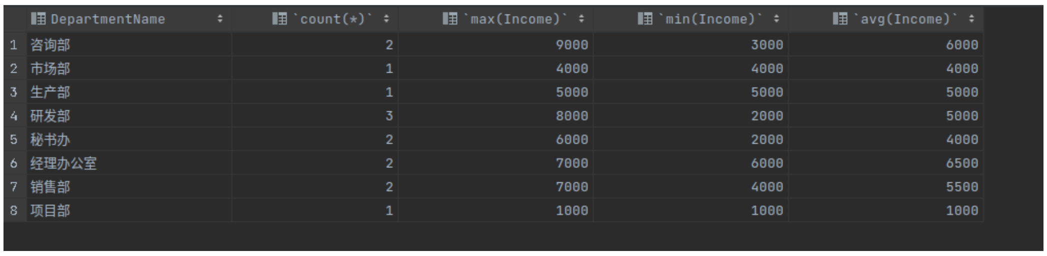
```sql
create table grading like Salary;
insert into grading select * from Salary;

alter table grading add grade int;

create procedure grading()
begin
    declare number int;
    declare i int;
    declare tem_income int;
    declare tem_id char(6);
    declare tem_grade int;
    set i = 0;
    select count(*) from grading into number;
    while(i < number) do
        select Income, EmployeeID into tem_income, tem_id from
grading limit i, 1;
        if(tem_income < 3000) then
            set tem_grade = 1;
        else
            set tem_grade = (tem_income - 1000) / 1000;
        end if;
        update grading set grade = tem_grade where EmployeeID =
tem_id;
        set i = i+1;
        set tem_grade = 0;
        end while;
end;

drop procedure grading;
call grading();

select *
from grading;
```



	EmployeeID	Income	Outcome	grade
1	0	2000	1000	1
2	1	3000	1000	2
3	10	3000	1000	2
4	11	4000	1000	3
5	12	5000	1000	4
6	13	6000	1000	5
7	14	7000	1000	6
8	2	4000	1000	3
9	3	5000	1000	4
10	4	6000	1000	5
11	5	7000	1000	6
12	6	8000	1000	7
13	7	9000	1000	8
14	8	1000	1000	1
15	9	2000	1000	1

b）编写一个函数，该函数的作用是统计公司各部门的员工人数和员工的最高收入、最低收入和平均收入（选做）

```sql
select DepartmentName, count(*), max(Income), min(Income),
avg(Income)
from Employees natural join Departments natural join Salary
group by DepartmentName;
```

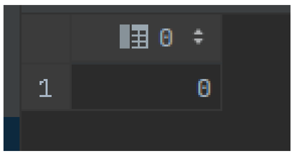


	DepartmentName	\`count(*)\`	\`max(Income)\`	\`min(Income)\`	\`avg(Income)\`
1	咨询部	2	9000	3000	6000
2	市场部	1	4000	4000	4000
3	生产部	1	5000	5000	5000
4	研发部	3	8000	2000	5000
5	秘书办	2	6000	2000	4000
6	经理办公室	2	7000	6000	6500
7	销售部	2	7000	4000	5500
8	项目部	1	1000	1000	1000

5）存储过程

练习 5.1： （a）创建一个存储过程，比较两个员工的实际收入，若前者比后者高就输出 0，否则输出 1

```
create procedure compare_income(name1 char(6), name2 char(6))
begin
    declare income1, income2 int;
    select Income into income1 from Salary where EmployeeID =
name1;
    select Income into income2 from Salary where EmployeeID =
name2;
    if income1 ⩾ income2 then
        select 0;
    else
        select 1;
    end if;
end;

drop procedure compare_income;
call compare_income('1', '0');

select *
from Salary;
```

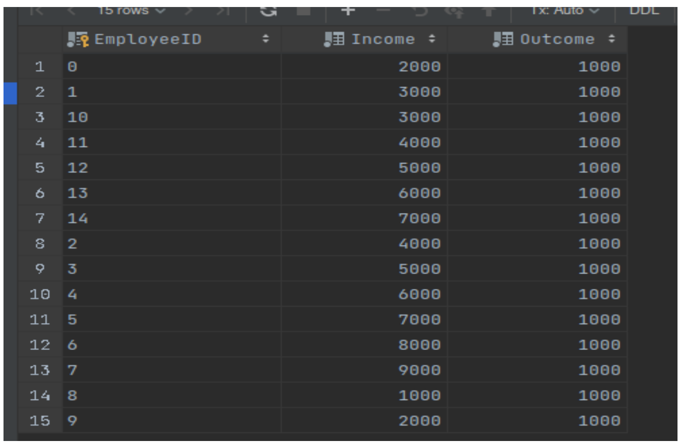


```
   0
1  0
```

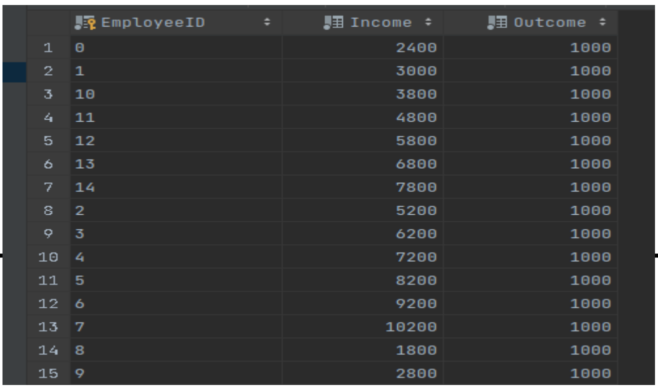
（b）创建一个存储过程，将每个人的收入提高 500。（如果是根据每个人的学历将收入提高，如大专及以下提高 400，本科提高 500，硕士提高 650，博士提高 800，又如何实现？选做）

```
create procedure add_income_by_education()
begin
    declare tem_education char(4);
    declare number, i int;
    declare tem_income int;
    declare tem_id char(6);
    set i = 0;
    select count(*) from Employees natural join Salary2 into
```

```
number;
    while(i < number) do
        select Income, EmployeeID, Education into tem_income,
tem_id, tem_education
        from Employees natural join Salary2 limit i, 1;
        case tem_education
            when '小学' then set tem_income = tem_income + 400;
            when '大学' then set tem_income = tem_income + 800;
            when '高中' then set tem_income = tem_income + 1200;
        end case;
        update Salary2 set Income = tem_income where EmployeeID =
tem_id;
        set i = i+1;
        end while;
end;

select *
from Salary2;

select *
from Salary;

call add_income_by_education();
```



	EmployeeID	Income	Outcome
1	0	2000	1000
2	1	3000	1000
3	10	3000	1000
4	11	4000	1000
5	12	5000	1000
6	13	6000	1000
7	14	7000	1000
8	2	4000	1000
9	3	5000	1000
10	4	6000	1000
11	5	7000	1000
12	6	8000	1000
13	7	9000	1000
14	8	1000	1000
15	9	2000	1000

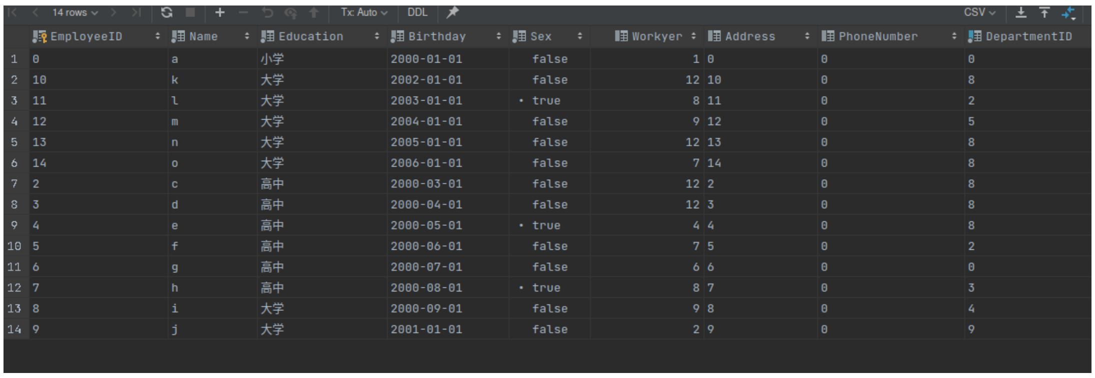


	EmployeeID	Income	Outcome
1	0	2400	1000
2	1	3000	1000
3	10	3800	1000
4	11	4800	1000
5	12	5800	1000
6	13	6800	1000
7	14	7800	1000
8	2	5200	1000
9	3	6200	1000
10	4	7200	1000
11	5	8200	1000
12	6	9200	1000
13	7	10200	1000
14	8	1800	1000
15	9	2800	1000

练习 5.2： （a）创建一个存储过程，要求一个员工的工作年份大于 10 年时将其转到经理办公室工作

```
create procedure cursor_move()
begin
    declare done int default false;
    declare tem_id char(6);
    declare tem_workyer char(3);
    declare need_id char(3);
    declare cursor_i cursor for select EmployeeID, Workyer from
Employees;
    declare continue handler for not found set done = true;

  select DepartmentID into need_id from Departments where
DepartmentName = '经理办公室';

  open cursor_i;
  read_loop: LOOP
    fetch cursor_i into tem_id, tem_workyer;
    if done then
      leave read_loop;
    END IF;
    if tem_workyer > 10 then
        update Employees1 set DepartmentID = need_id where
EmployeeID = tem_id;
    end if;
  end loop;
  close cursor_i;
end;

create table Employees1 like Employees;
insert into Employees1 select * from Employees;

select *
from Employees1;
```

```sql
call cursor_move();
```



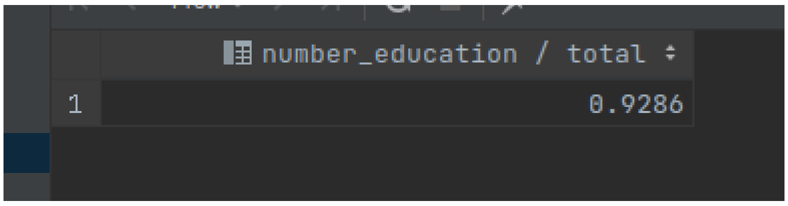
14 rows · Tx: Auto · DDL · CSV

	EmployeeID	Name	Education	Birthday	Sex	Workyer	Address	PhoneNumber	DepartmentID
1	0	a	小学	2000-01-01	false	1	0	0	0
2	10	k	大学	2002-01-01	false	12	10	0	8
3	11	l	大学	2003-01-01	true	8	11	0	2
4	12	m	大学	2004-01-01	false	9	12	0	5
5	13	n	大学	2005-01-01	false	12	13	0	8
6	14	o	大学	2006-01-01	false	7	14	0	8
7	2	c	高中	2000-03-01	false	12	2	0	8
8	3	d	高中	2000-04-01	false	12	3	0	8
9	4	e	高中	2000-05-01	true	4	4	0	8
10	5	f	高中	2000-06-01	false	7	5	0	2
11	6	g	高中	2000-07-01	false	6	6	0	0
12	7	h	高中	2000-08-01	true	8	7	0	3
13	8	i	大学	2000-09-01	false	9	8	0	4
14	9	j	大学	2001-01-01	false	2	9	0	9

(b）创建一个存储过程，使用游标计算本科及以上学历的员工在总员工中所占比例。（选做）

```sql
create procedure compute()
begin
    declare done int default false;
    declare tem_education char(4);
    declare total int;
    declare number_education int;
    declare cursor_education cursor for select Education from
Employees;
    declare continue handler for not found set done = true;

    set number_education = 0;
    select count(*) into total from Employees;
    open cursor_education;
    label: loop
    fetch cursor_education into tem_education;
    if done then
        leave label;
    end if;
    if tem_education ≠ '小学' then
```

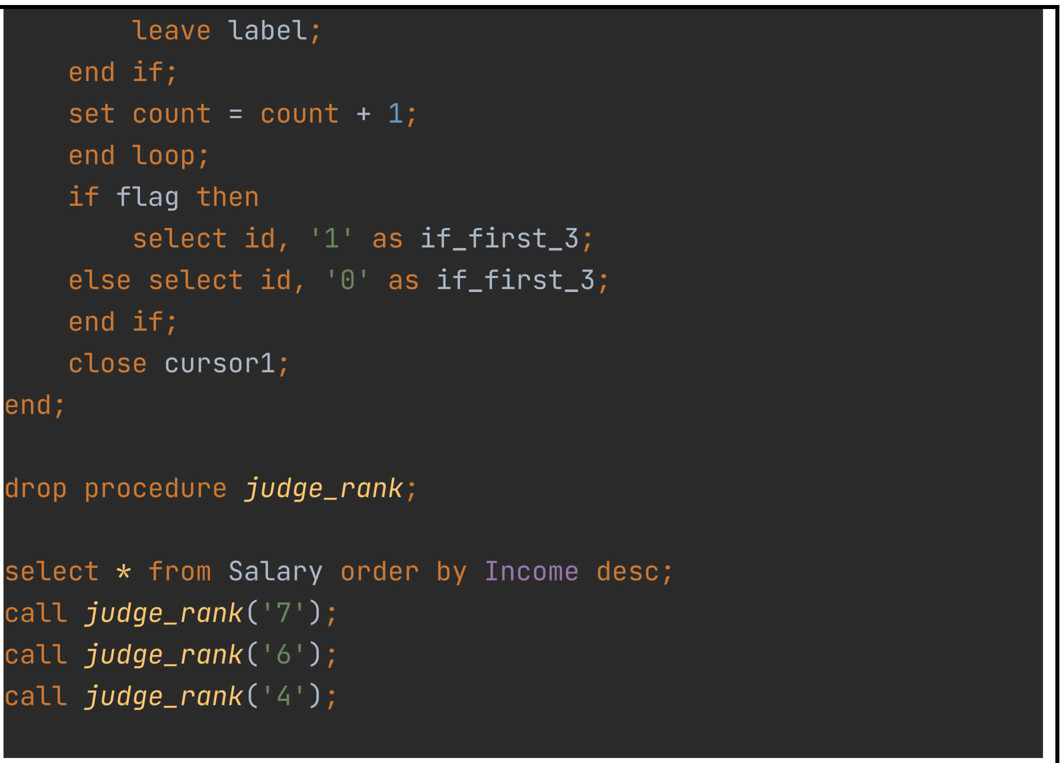
```
        set number_education = number_education + 1;
    end if;
    end loop;
    close cursor_education;
    select number_education / total;
end;

call compute();
```



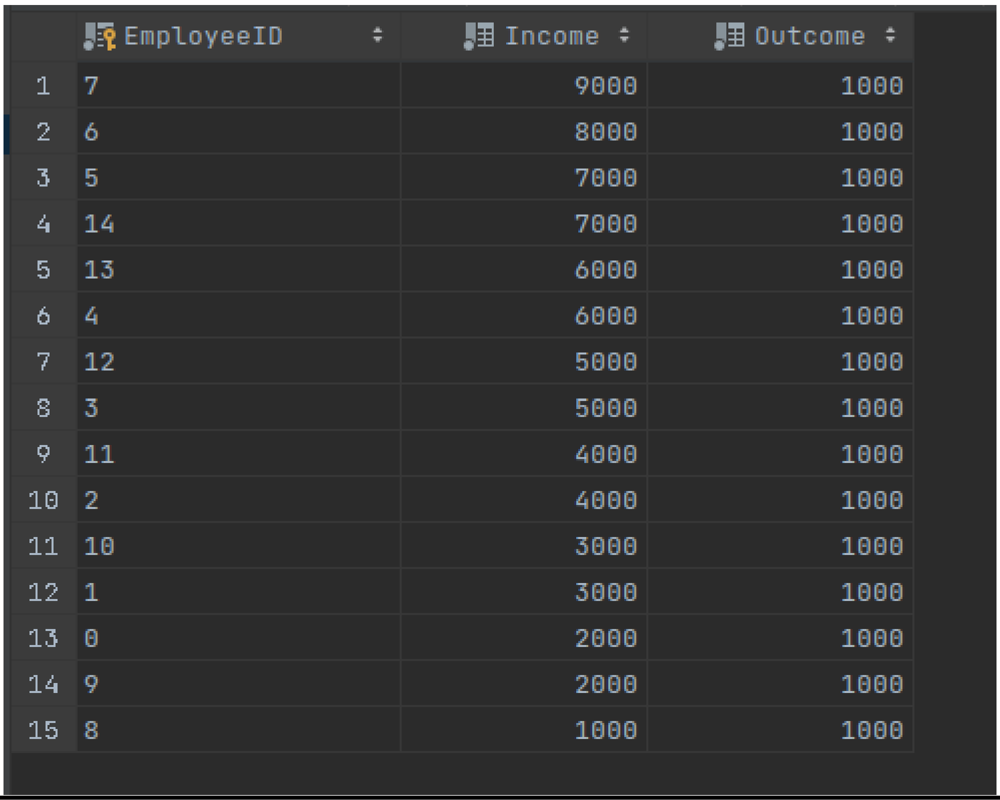
```
   number_education / total
1                    0.9286
```

（c）创建存储过程，使用游标确定一个员工的实际收入是否排在前三名，结果为 1 表示是，结果为 0 表示否。（选做）

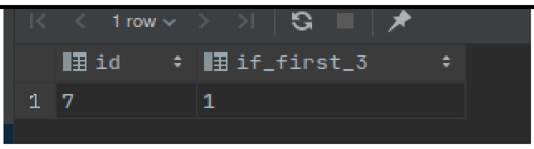
```
create procedure judge_rank(id char(6))
begin
    declare tem_id char(6);
    declare count int;
    declare flag int;
    declare cursor1 cursor for select EmployeeID from Salary
order by Income desc;
    set count = 0;
    set flag = 0;

    open cursor1;
    label: loop
        fetch cursor1 into tem_id;
    if count = 3 then
        leave label;
    end if;
    if tem_id = id then
        set flag = 1;
```



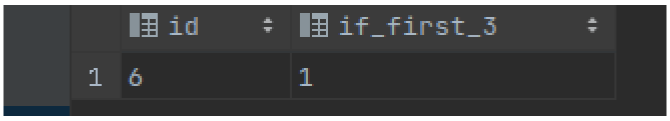
```
        leave label;
    end if;
    set count = count + 1;
    end loop;
    if flag then
        select id, '1' as if_first_3;
    else select id, '0' as if_first_3;
    end if;
    close cursor1;
end;

drop procedure judge_rank;

select * from Salary order by Income desc;
call judge_rank('7');
call judge_rank('6');
call judge_rank('4');
```



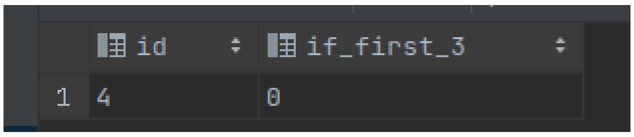
	EmployeeID	Income	Outcome
1	7	9000	1000
2	6	8000	1000
3	5	7000	1000
4	14	7000	1000
5	13	6000	1000
6	4	6000	1000
7	12	5000	1000
8	3	5000	1000
9	11	4000	1000
10	2	4000	1000
11	10	3000	1000
12	1	3000	1000
13	0	2000	1000
14	9	2000	1000
15	8	1000	1000



```
   id   if_first_3
1  7    1
```



```
   id   if_first_3
1  6    1
```



```
   id   if_first_3
1  4    0
```

3．总结与问题分析（100 字以上）

语法难找，还有 mysql 有一些语法不支持，像自定义数据类型，也没有 print 函数，只能找些代替品了，不是很全。

函数的定义必须要有返回值，一律用 procedure 代替，影响不大。而且函数里面不能 select 输出，报什么 value set 的错误。

在浏览一个表时，除了用 cursor 之外，limit 也挺好使的，加个循环。

Limit I 1， 从 I 行起，查一行 ，那在插入语句时怎样插入指定行的顺序呢，实现排序之类的。

在网上查的过程中，有一个 label，好像挺重要的样子，还有什么 loop 之类的。