# 题目：PL-SQL 经典试题

0. 准备工作:

```
set serveroutput on
```

hellowrold 程序

```
begin
dbms_output.put_line('hello  world');
end;
/
```

[语法格式]

```
--declare
   --声明的变量、类型、游标
begin
   --程序的执行部分（类似于 java 里的 main()方法）
   dbms_output.put_line('helloworld');
--exception
   --针对 begin 块中出现的异常，提供处理的机制
   --when .... then ...
   --when    .... then ...
end;
```

**********

基本语法

**********

1. 使用一个变量

```
declare
   --声明一个变量
   v_name varchar2(25);
begin
   --通过  select ... into ...  语句为变量赋值
  select last_name into v_name
  from employees
  where employee_id = 186;

  --  打印变量的值
  dbms_output.put_line(v_name);
end;
```

2. 使用多个变量

```
declare
   --声明变量
   v_name varchar2(25);
   v_email varchar2(25);
   v_salary number(8, 2);
   v_job_id varchar2(10);
begin
   --通过  select ... into ...  语句为变量赋值
   --被赋值的变量与 SELECT 中的列名要一一对应
  select last_name, email, salary, job_id into v_name, v_email, v_salary, v_job_id
  from employees
  where employee_id = 186;

  -- 打印变量的值
  dbms_output.put_line(v_name  || ', ' || v_email || ', ' ||    v_salary || ', ' ||    v_job_id);
end;
```

---

记录类型

---

3.1 自定义记录类型

```
declare
   --定义一个记录类型
   type customer_type is record(
      v_cust_name varchar2(20),
      v_cust_id number(10));

   --声明自定义记录类型的变量
   v_customer_type customer_type;
begin
   v_customer_type.v_cust_name := '刘德华';
   v_customer_type.v_cust_id := 1001;

   dbms_output.put_line(v_customer_type.v_cust_name||','||v_customer_type.v_cust_id);
end;
```

3.2 自定义记录类型

```
declare
   --定义一个记录类型
   type emp_record is record(
      v_name varchar2(25),
      v_email varchar2(25),
```

```
    v_salary number(8, 2),
    v_job_id varchar2(10));

  --声明自定义记录类型的变量
  v_emp_record emp_record;
begin
  --通过 select ... into ... 语句为变量赋值
 select last_name, email, salary, job_id into v_emp_record
 from employees
 where employee_id = 186;

 -- 打印变量的值
 dbms_output.put_line(v_emp_record.v_name || ', ' || v_emp_record.v_email || ', ' ||
v_emp_record.v_salary || ',' ||  v_emp_record.v_job_id);
end;
```

4. 使用 %type 定义变量，动态的获取数据的声明类型

```
declare
  --定义一个记录类型
  type emp_record is record(
    v_name employees.last_name%type,
    v_email employees.email%type,
    v_salary employees.salary%type,
    v_job_id employees.job_id%type);

  --声明自定义记录类型的变量
  v_emp_record emp_record;
begin
  --通过 select ... into ... 语句为变量赋值
 select last_name, email, salary, job_id into v_emp_record
 from employees
 where employee_id = 186;

 -- 打印变量的值
 dbms_output.put_line(v_emp_record.v_name || ', ' || v_emp_record.v_email || ', ' ||
v_emp_record.v_salary || ',' ||  v_emp_record.v_job_id);
end;
```

5. 使用 %rowtype

```
declare
--声明一个记录类型的变量
```

```
   v_emp_record employees%rowtype;
begin
   --通过  select ... into ...  语句为变量赋值
  select * into v_emp_record
  from employees
  where employee_id = 186;

  -- 打印变量的值
  dbms_output.put_line(v_emp_record.last_name  ||  ',  '  ||  v_emp_record.email  ||  ',  '  ||
v_emp_record.salary || ', ' ||    v_emp_record.job_id    || ', ' || v_emp_record.hire_date);
end;
```

6.1 赋值语句：通过变量实现查询语句

```
declare
   v_emp_record employees%rowtype;
   v_employee_id employees.employee_id%type;
begin
   --使用赋值符号位变量进行赋值
   v_employee_id := 186;

   --通过  select ... into ...  语句为变量赋值
  select * into v_emp_record
  from employees
  where employee_id = v_employee_id;

  -- 打印变量的值
  dbms_output.put_line(v_emp_record.last_name  || ', ' || v_emp_record.email || ', ' ||
v_emp_record.salary || ', ' ||    v_emp_record.job_id    || ', ' || v_emp_record.hire_date);
end;
```

6.2　通过变量实现 DELETE、INSERT、UPDATE 等操作

```
declare
   v_emp_id employees.employee_id%type;

begin
   v_emp_id := 109;
   delete from employees
   where employee_id = v_emp_id;
   --commit;
end;
```

**********

**********

---

条件判断

---

7. 使用 IF ... THEN ... ELSIF ... THEN ...ELSE ... END IF;

要求: 查询出 150 号 员工的工资, 若其工资大于或等于 10000 则打印 'salary >= 10000'; 若在 5000 到 10000 之间, 则打印 '5000<= salary < 10000'; 否则打印 'salary < 5000'

(方法一)

```
declare
   v_salary employees.salary%type;
begin
   --通过 select ... into ... 语句为变量赋值
  select salary into v_salary
  from employees
  where employee_id = 150;

  dbms_output.put_line('salary: ' || v_salary);

  -- 打印变量的值
  if v_salary >= 10000 then
       dbms_output.put_line('salary >= 10000');
  elsif v_salary >= 5000 then
       dbms_output.put_line('5000 <= salary < 10000');
  else
       dbms_output.put_line('salary < 5000');
  end if;
```

(方法二)

```
declare
        v_emp_name employees.last_name%type;
        v_emp_sal employees.salary%type;
        v_emp_sal_level varchar2(20);
begin
        select last_name,salary into v_emp_name,v_emp_sal from employees where employee_id
= 150;

        if(v_emp_sal >= 10000) then v_emp_sal_level := 'salary >= 10000';
        elsif(v_emp_sal >= 5000) then v_emp_sal_level := '5000<= salary < 10000';
        else v_emp_sal_level := 'salary < 5000';
        end if;

        dbms_output.put_line(v_emp_name||','||v_emp_sal||','||v_emp_sal);
end;
```

7+ 使用 CASE ... WHEN ... THEN ...ELSE ... END 完成上面的任务

```
declare
        v_sal employees.salary%type;
        v_msg varchar2(50);
begin
        select salary into v_sal
        from employees
        where employee_id = 150;

        --case 不能向下面这样用
        /*
        case v_sal when salary >= 10000 then v_msg := '>=10000'
                        when salary >= 5000 then v_msg := '5000<= salary < 10000'
                        else v_msg := 'salary < 5000'
        end;
        */

        v_msg :=
                case trunc(v_sal / 5000)
                        when 0 then 'salary < 5000'
                        when 1 then '5000<= salary < 10000'
                        else 'salary >= 10000'
                end;

        dbms_output.put_line(v_sal ||','||v_msg);
end;
```

8. 使用 CASE ... WHEN ... THEN ... ELSE ... END;
要求: 查询出 122 号员工的 JOB_ID, 若其值为 'IT_PROG', 则打印 'GRADE: A';
'AC_MGT', 打印 'GRADE B',
'AC_ACCOUNT', 打印 'GRADE C';
否则打印 'GRADE D'

```
declare
        --声明变量
        v_grade char(1);
        v_job_id employees.job_id%type;
begin
        select job_id into v_job_id
        from employees
        where employee_id = 122;

        dbms_output.put_line('job_id:  ' || v_job_id);
```

```
        --根据 v_job_id 的取值, 利用 case 字句为 v_grade 赋值
        v_grade :=
                case v_job_id when 'IT_PROG' then 'A'
                              when 'AC_MGT' then 'B'
                              when 'AC_ACCOUNT' then 'C'
                              else 'D'
                 end;

        dbms_output.put_line('GRADE: ' || v_grade);
end;
```

---

循环结构

---

9. 使用循环语句打印 1 - 100.（三种方式）

```
1).   LOOP ... EXIT WHEN ... END LOOP
declare
        --初始化条件
        v_i number(3) := 1;
begin
        loop
        --循环体
          dbms_output.put_line(v_i);
                                                    --循环条件
          exit when v_i = 100;
                                                    --迭代条件
          v_i := v_i + 1;
        end loop;
end;

2). WHILE ... LOOP ... END LOOP
declare
        --初始化条件
        v_i number(3) := 1;
begin
        --循环条件
        while v_i <= 100 loop
                                                        --循环体
                dbms_output.put_line(v_i);
                                                        --迭代条件
                v_i := v_i + 1;
        end loop;
end;
```

3).

```
begin
        for i in 1 .. 100 loop
                dbms_output.put_line(i);
        end loop;
end;
```

10. 综合使用 if, while 语句, 打印 1 - 100 之间的所有素数(素数: 有且仅用两个正约数的整数, 2, 3, 5, 7, 11, 13, ...).

```
declare
   v_flag number(1):=1;
   v_i number(3):=2;
   v_j number(2):=2;
begin

   while (v_i<=100) loop
            while v_j <= sqrt(v_i) loop
                    if (mod(v_i,v_j)=0) then v_flag:= 0;end if;
                    v_j :=v_j +1;
            end loop;
            if(v_flag=1) then dbms_output.put_line(v_i);end  if;

            v_flag :=1;
            v_j := 2;
            v_i :=v_i +1;
     end loop;

end;
```

(法二)使用 for 循环实现 1-100 之间的素数的输出

```
declare
   --标记值, 若为 1 则是素数, 否则不是
   v_flag number(1) := 0;
begin
     for i in 2 .. 100 loop

        v_flag := 1;

        for j in 2 .. sqrt(i) loop
            if i mod j = 0 then
                v_flag := 0;
            end if;
```

```
        end loop;

        if v_flag = 1 then
            dbms_output.put_line(i);
        end if;

    end loop;
end;
```

11. 使用 goto

```
declare
  --标记值，若为 1 则是素数，否则不是
  v_flag number(1) := 0;
begin
    for i in 2 .. 100 loop
        v_flag := 1;

        for j in 2 .. sqrt(i) loop
            if i mod j = 0 then
                v_flag := 0;
                goto label;
            end if;
        end loop;

        <<label>>
        if v_flag = 1 then
            dbms_output.put_line(i);
        end if;

    end loop;
end;
```

11+.打印 1——100 的自然数，当打印到 50 时，跳出循环，输出“打印结束”

(方法一)

```
begin
  for i in 1..100 loop
      dbms_output.put_line(i);
      if(i = 50) then
      goto label;
      end if;
  end loop;

      <<label>>
      dbms_output.put_line('打印结束');
```

```
end;
(方法二)
begin
   for i in 1..100 loop
         dbms_output.put_line(i);
         if(i mod 50 = 0) then dbms_output.put_line('打印结束');
         exit;
         end if;
   end loop;
end;
```

**********

游标的使用

**********

12.1 使用游标

要求: 打印出 80 部门的所有的员工的工资:salary: xxx

```
declare
   --1. 定义游标
   cursor salary_cursor is select salary from employees where department_id = 80;
   v_salary employees.salary%type;
begin
  --2. 打开游标
  open salary_cursor;

  --3. 提取游标
  fetch salary_cursor into v_salary;

  --4. 对游标进行循环操作: 判断游标中是否有下一条记录
while salary_cursor%found loop
         dbms_output.put_line('salary: ' || v_salary);
         fetch salary_cursor into v_salary;
end loop;

  --5. 关闭游标
  close   salary_cursor;
end;
```

12.2 使用游标

要求: 打印出 80 部门的所有的员工的工资: Xxx 's salary is: xxx

```
declare
   cursor sal_cursor is select salary ,last_name from employees where department_id = 80;
   v_sal number(10);
```

```
   v_name varchar2(20);
begin
   open sal_cursor;

   fetch sal_cursor into v_sal,v_name;

   while sal_cursor%found  loop
            dbms_output.put_line(v_name||'`s  salary is '||v_sal);
            fetch sal_cursor into v_sal,v_name;
   end loop;

   close sal_cursor;

end;
```

13. 使用游标的练习:
打印出 manager_id 为 100 的员工的 last_name, email, salary 信息(使用游标, 记录类型)

```
declare
            --声明游标
            cursor emp_cursor is select last_name, email, salary from employees where
manager_id = 100;

            --声明记录类型
            type emp_record is record(
                  name employees.last_name%type,
                  email employees.email%type,
                  salary employees.salary%type
            );

            -- 声明记录类型的变量
            v_emp_record emp_record;
begin
            --打开游标
            open emp_cursor;

            --提取游标
            fetch emp_cursor into v_emp_record;

            --对游标进行循环操作
            while emp_cursor%found  loop
                  dbms_output.put_line(v_emp_record.name || ', ' || v_emp_record.email
|| ', ' || v_emp_record.salary );
                  fetch emp_cursor into v_emp_record;
```

```
        end loop;

        --关闭游标
        close emp_cursor;
end;
```

(法二：使用 for 循环)

```
declare

    cursor emp_cursor is
    select last_name,email,salary
    from employees
    where manager_id = 100;

begin


    for v_emp_record in emp_cursor loop

dbms_output.put_line(v_emp_record.last_name||','||v_emp_record.email||','||v_emp_record.
salary);
    end loop;
end;
```

14. 利用游标，调整公司中员工的工资:

| 工资范围 | 调整基数 |
| --- | --- |
| 0 - 5000 | 5% |
| 5000 - 10000 | 3% |
| 10000 - 15000 | 2% |
| 15000 - | 1% |

```
declare
    --定义游标
    cursor emp_sal_cursor is select salary, employee_id from employees;

    --定义基数变量
    temp number(4, 2);

    --定义存放游标值的变量
    v_sal employees.salary%type;
    v_id employees.employee_id%type;
begin
    --打开游标
    open emp_sal_cursor;
```

```
    --提取游标
    fetch emp_sal_cursor into v_sal, v_id;

    --处理游标的循环操作
    while emp_sal_cursor%found loop
        --判断员工的工资，执行 update 操作
        --dbms_output.put_line(v_id || ': ' || v_sal);

        if v_sal <= 5000 then
            temp := 0.05;
        elsif v_sal<= 10000 then
            temp := 0.03;
        elsif v_sal <= 15000 then
            temp := 0.02;
        else
            temp := 0.01;
        end if;

        --dbms_output.put_line(v_id || ': ' || v_sal || ', ' || temp);
        update employees set salary = salary * (1 + temp) where employee_id = v_id;

        fetch emp_sal_cursor into v_sal, v_id;
    end loop;
    --关闭游标
    close emp_sal_cursor;
end;
```

使用 SQL 中的 decode 函数

```
update employees set salary = salary * (1 + (decode(trunc(salary/5000), 0, 0.05,
                                                                        1, 0.03,
                                                                        2, 0.02,
                                                                        0.01)))
```

15. 利用游标 for 循环完成 14.

```
declare
    --定义游标
    cursor emp_sal_cursor is select salary, employee_id id from employees;

    --定义基数变量
    temp number(4, 2);
begin
```

```
    --处理游标的循环操作
    for c in emp_sal_cursor loop
        --判断员工的工资，执行 update 操作
        --dbms_output.put_line(c.employee_id || ': ' || c.salary);

        if c.salary <= 5000 then
            temp := 0.05;
        elsif c.salary <= 10000 then
            temp := 0.03;
        elsif c.salary <= 15000 then
            temp := 0.02;
        else
            temp := 0.01;
        end if;

        --dbms_output.put_line(v_id || ': ' || v_sal || ', ' || temp);
        update employees set salary = salary * (1 + temp) where employee_id = c.id;
    end loop;
end;
```

16*. 带参数的游标

```
declare
    --定义游标
    cursor emp_sal_cursor(dept_id number, sal number) is
        select salary + 1000 sal, employee_id id
        from employees
        where department_id = dept_id and salary > sal;

    --定义基数变量
    temp number(4, 2);
begin
    --处理游标的循环操作
    for c in emp_sal_cursor(sal => 4000, dept_id => 80) loop
        --判断员工的工资，执行 update 操作
        --dbms_output.put_line(c.id || ': ' || c.sal);

        if c.sal <= 5000 then
            temp := 0.05;
        elsif c.sal <= 10000 then
            temp := 0.03;
        elsif c.sal <= 15000 then
            temp := 0.02;
        else
```

```
            temp := 0.01;
        end if;

        dbms_output.put_line(c.sal || ': ' || c.id || ', ' || temp);
        --update employees set salary = salary * (1 + temp) where employee_id = c.id;
    end loop;
end;
```

17. 隐式游标：更新指定员工 salary(涨工资 10)，如果该员工没有找到，则打印”查无此人”信息

```
begin
        update employees set salary = salary + 10 where employee_id = 1005;

        if sql%notfound then
            dbms_output.put_line('查无此人!');
        end if;
end;
```

**********

异常处理

**********

[预定义异常]

```
declare

  v_sal employees.salary%type;
begin
  select salary into v_sal
  from employees
  where employee_id >100;

  dbms_output.put_line(v_sal);

exception
  when Too_many_rows then dbms_output.put_line('输出的行数太多了');
end;
```

[非预定义异常]

```
declare

  v_sal employees.salary%type;
  --声明一个异常
  delete_mgr_excep exception;
  --把自定义的异常和 oracle 的错误关联起来
```

```
   PRAGMA EXCEPTION_INIT(delete_mgr_excep,-2292);
begin
   delete from employees
   where employee_id = 100;

   select salary into v_sal
   from employees
   where employee_id >100;

   dbms_output.put_line(v_sal);

exception
   when Too_many_rows then dbms_output.put_line('输出的行数太多了');
   when delete_mgr_excep then dbms_output.put_line('Manager 不能直接被删除');
end;

[用户自定义异常]
declare

   v_sal employees.salary%type;
   --声明一个异常
   delete_mgr_excep exception;
   --把自定义的异常和 oracle 的错误关联起来
   PRAGMA EXCEPTION_INIT(delete_mgr_excep,-2292);

   --声明一个异常
   too_high_sal exception;
begin

   select salary into v_sal
   from employees
   where employee_id =100;

   if v_sal > 1000 then
       raise too_high_sal;
   end if;

   delete from employees
   where employee_id = 100;

   dbms_output.put_line(v_sal);

exception
   when Too_many_rows then dbms_output.put_line('输出的行数太多了');
```

```
  when delete_mgr_excep then dbms_output.put_line('Manager 不能直接被删除');
  --处理异常
  when too_high_sal then dbms_output.put_line('工资过高了');
end;
```

18. 异常的基本程序:
通过 select ... into ... 查询某人的工资，若没有查询到，则输出 "未找到数据"

```
declare
  --定义一个变量
  v_sal employees.salary%type;
begin
  --使用 select ... into ... 为 v_sal 赋值
  select salary into v_sal from employees where employee_id = 1000;
  dbms_output.put_line('salary:    ' || v_sal);
exception
  when No_data_found then
       dbms_output.put_line('未找到数据');
end;
```

或

```
declare
  --定义一个变量
  v_sal employees.salary%type;
begin
  --使用 select ... into ... 为 v_sal 赋值
  select salary into v_sal from employees;
  dbms_output.put_line('salary:    ' || v_sal);
exception
  when No_data_found then
       dbms_output.put_line('未找到数据!');
  when Too_many_rows then
       dbms_output.put_line('数据过多!');
end;
```

19. 更新指定员工工资，如工资小于 300，则加 100；对 NO_DATA_FOUND 异常，TOO_MANY_ROWS 进行处理.

```
declare
    v_sal employees.salary%type;
begin
    select salary into v_sal from employees where employee_id = 100;

    if(v_sal < 300) then update employees set salary = salary + 100 where employee_id = 100;
```

```
    else dbms_output.put_line('工资大于 300');
    end if;
exception
    when no_data_found then dbms_output.put_line('未找到数据');
     when too_many_rows then dbms_output.put_line('输出的数据行太多');
end;
```

20. 处理非预定义的异常处理: "违反完整约束条件"

```
declare
  --1. 定义异常
  temp_exception exception;

  --2. 将其定义好的异常情况，与标准的 ORACLE 错误联系起来，使用 EXCEPTION_INIT 语句
  PRAGMA EXCEPTION_INIT(temp_exception, -2292);
begin
  delete from employees where employee_id = 100;

exception
  --3. 处理异常
  when temp_exception then
        dbms_output.put_line('违反完整性约束!');
end;
```

21. 自定义异常: 更新指定员工工资，增加 100；若该员工不存在则抛出用户自定义异常: no_result

```
declare
  --自定义异常
  no_result exception;
begin
  update employees set salary = salary + 100 where employee_id = 1001;

  --使用隐式游标, 抛出自定义异常
  if sql%notfound then
      raise no_result;
  end if;

exception

  --处理程序抛出的异常
  when no_result then
      dbms_output.put_line('更新失败');
```

```
end;
```

**********

存储函数和过程

**********

[存储函数：有返回值，创建完成后，通过 select function() from dual;执行]
[存储过程：由于没有返回值，创建完成后，不能使用 select 语句，只能使用 pl/sql 块执行]

[格式]

```
--函数的声明(有参数的写在小括号里)
create or replace function func_name(v_param varchar2)
--返回值类型
return varchar2
is
--PL/SQL 块变量、记录类型、游标的声明(类似于前面的 declare 的部分)
begin
--函数体(可以实现增删改查等操作，返回值需要 return)
        return 'helloworld'|| v_param;
end;
```

22.1 函数的 helloworld: 返回一个 "helloworld" 的字符串

```
create or replace function hello_func
return varchar2
is
begin
        return 'helloworld';
end;
```

执行函数

```
begin
    dbms_output.put_line(hello_func());
end;
```

或者： select hello_func() from dual;

22.2 返回一个"helloworld: atguigu"的字符串，其中 atguigu 由执行函数时输入。

```
--函数的声明(有参数的写在小括号里)
create or replace function hello_func(v_logo varchar2)
--返回值类型
return varchar2
is
--PL/SQL 块变量的声明
```

```
begin
--函数体
        return 'helloworld'|| v_logo;
end;
```

22.3 创建一个存储函数，返回当前的系统时间

```
create or replace function func1
return date
is
--定义变量
v_date date;
begin
-函数体
--v_date := sysdate;
    select sysdate into v_date from dual;
        dbms_output.put_line('我是函数哦');

        return v_date;
end;
```

执行法 1：

```
select func1 from dual;
```

执行法 2：

```
declare
  v_date date;
begin
  v_date := func1;
  dbms_output.put_line(v_date);
end;
```

23. 定义带参数的函数：两个数相加

```
create or replace function add_func(a number, b number)
return number
is
begin
        return (a + b);
end;
```

执行函数

```
begin
    dbms_output.put_line(add_func(12, 13));
end;
```

或者

```
select add_func(12,13) from dual;
```

24. 定义一个函数: 获取给定部门的工资总和, 要求:部门号定义为参数, 工资总额定义为返回值.

```
create or replace function sum_sal(dept_id number)
        return number
        is

        cursor sal_cursor is select salary from employees where department_id = dept_id;
        v_sum_sal number(8) := 0;
begin
        for c in sal_cursor loop
            v_sum_sal := v_sum_sal + c.salary;
        end loop;

        --dbms_output.put_line('sum salary: ' || v_sum_sal);
        return v_sum_sal;
end;
```

执行函数

```
begin
    dbms_output.put_line(sum_sal(80));
end;
```

25. 关于 OUT 型的参数: 因为函数只能有一个返回值, PL/SQL 程序可以通过 OUT 型的参数实现有多个返回值

要求: 定义一个函数: 获取给定部门的工资总和 和 该部门的员工总数(定义为 OUT 类型的参数).
要求: 部门号定义为参数, 工资总额定义为返回值.

```
create or replace function sum_sal(dept_id number, total_count out number)
        return number
        is

        cursor sal_cursor is select salary from employees where department_id = dept_id;
        v_sum_sal number(8) := 0;
begin
        total_count := 0;

        for c in sal_cursor loop
```

```
            v_sum_sal := v_sum_sal + c.salary;
            total_count := total_count + 1;
        end loop;

        --dbms_output.put_line('sum  salary: ' || v_sum_sal);
        return v_sum_sal;
end;
```

执行函数:

```
delare
   v_total number(3) := 0;

begin
      dbms_output.put_line(sum_sal(80,  v_total));
      dbms_output.put_line(v_total);
end;
```

26*. 定义一个存储过程: 获取给定部门的工资总和(通过 out 参数), 要求:部门号和工资总额定义为参数

```
create or replace procedure sum_sal_procedure(dept_id  number, v_sum_sal out number)
        is

        cursor sal_cursor is select salary from employees where department_id = dept_id;
begin
        v_sum_sal := 0;

        for c in sal_cursor loop
            --dbms_output.put_line(c.salary);
            v_sum_sal := v_sum_sal + c.salary;
        end loop;

        dbms_output.put_line('sum  salary: ' || v_sum_sal);
end;
[执行]
declare
      v_sum_sal number(10) := 0;
begin
      sum_sal_procedure(80,v_sum_sal);
end;
```

27*. 自定义一个存储过程完成以下操作:
对给定部门(作为输入参数)的员工进行加薪操作, 若其到公司的时间在

(? , 95) 期间, 为其加薪 %5
[95 , 98) %3
[98, ?) %1
得到以下返回结果: 为此次加薪公司每月需要额外付出多少成本(定义一个 OUT 型的输出参数).

```
create or replace procedure add_sal_procedure(dept_id number, temp out number)

is

        cursor sal_cursor is select employee_id id, hire_date hd, salary sal from employees
where department_id = dept_id;
        a number(4, 2) := 0;
begin
        temp := 0;

        for c in sal_cursor loop
            a := 0;

            if c.hd < to_date('1995-1-1', 'yyyy-mm-dd') then
                a := 0.05;
            elsif c.hd < to_date('1998-1-1', 'yyyy-mm-dd') then
                a := 0.03;
            else
                a := 0.01;
            end if;

            temp := temp + c.sal * a;
            update employees set salary = salary * (1 + a) where employee_id = c.id;
        end loop;
end;
```

**********

触发器

**********

一个 helloworld 级别的触发器

```
create or replace trigger hello_trigger
after
update on employees
--for each row
begin
    dbms_output.put_line('hello...');
    --dbms_output.put_line('old.salary:'||  :OLD.salary||',new.salary'||:NEW.salary);
end;
```

然后执行：update employees set salary = salary + 1000;

28. 触发器的 helloworld: 编写一个触发器，在向 emp 表中插入记录时，打印 'helloworld'

```
create or replace trigger emp_trigger
after
insert on emp
for each row
begin
        dbms_output.put_line('helloworld');
end;
```

29. 行级触发器: 每更新 employees 表中的一条记录，都会导致触发器执行

```
create or replace trigger employees_trigger
after
update on employees
for each row
begin
        dbms_output.put_line('修改了一条记录!');
end;
```

语句级触发器: 一个 update/delete/insert 语句只使触发器执行一次

```
create or replace trigger employees_trigger
after
update on employees
begin
        dbms_output.put_line('修改了一条记录!');
end;
```

30. 使用 :new, :old 修饰符

```
create or replace trigger employees_trigger
after
update on employees
for each row
begin
        dbms_output.put_line('old  salary: ' || :old.salary || ', new salary: ' || :new.salary);
end;
```

31. 编写一个触发器，在对 my_emp 记录进行删除的时候，在 my_emp_bak 表中备份对应的记录

1). 准备工作:

```
create table my_emp as select employee_id id, last_name name, salary sal from employees;

create table my_emp_bak as select employee_id id, last_name name, salary sal from employees
where 1 = 2
```

2).

```
create or replace trigger bak_emp_trigger
        before delete on my_emp
        for each row

begin
        insert into my_emp_bak values(:old.id, :old.name, :old.sal);
end;
```