Title: PL SQL Programs

Author: S. Santhosh Kumar

SQL Programs and Outputs

PROGRAM 1: Calculate the Incentive of an Employee with ID 110

```
set serveroutput on;
declare
    v_emp_id employees.employee_id%type := 110;
    v_incentive number(7,2);
begin
    -- Assume the incentive is 10% of salary
    select salary * 0.1 into v_incentive
    from employees
    where employee_id = v_emp_id;

    dbms_output.put_line('The incentive for employee ID ' || v_emp_id || '
is: ' || v_incentive);
end;
/
```

Output:

```
Copy code
The incentive for employee ID 110 is: [calculated_value]
```

PROGRAM 2: Show Invalid Case-Insensitive Reference to a Quoted and Unquoted User-Defined Identifier

```
set serveroutput on;
declare
    "MyIdentifier" number := 100;
    myidentifier number := 200;
begin
    -- This will cause an error due to case sensitivity in quoted
identifiers
    dbms_output.put_line('Value of MyIdentifier: ' || MyIdentifier);  --
Invalid reference
end;
/
```

Output:

```
ORA-06550: line 5, column 52:
PLS-00201: identifier 'MYIDENTIFIER' must be declared
ORA-06550: line 5, column 5:
PL/SQL: Statement ignored
```

PROGRAM 3: Adjust the Salary of Employee with ID 122

```
set serveroutput on;
declare
    v_emp_id employees.employee_id%type := 122;
    v_salary employees.salary%type;
    v_new_salary employees.salary%type;
begin
    -- Fetch current salary
    select salary into v_salary
    from employees
    where employee_id = v_emp_id;

    -- Assume salary adjustment is a 10% increase
    v_new_salary := v_salary * 1.10;

    -- Update the salary
    update employees
    set salary = v_new_salary
    where employee_id = v_emp_id;

    dbms_output.put_line('Salary updated for employee ID ' || v_emp_id || '
to ' || v_new_salary);
end;
/
```

Output:

```
Salary updated for employee ID 122 to [new_salary_value]
```

PROGRAM 4: Procedure Using "IS [NOT] NULL Operator"

```
create or replace procedure check_null_and_condition
is
    v_value1 number := 10;
    v_value2 number := null;
begin
    if v_value1 is not null and v_value2 is null then
        dbms_output.put_line('Both conditions are met.');
    else
        dbms_output.put_line('Conditions are not met.');
    end if;
end;
/
```

Output:

```
Both conditions are met.
```

PROGRAM 5: Describe the Usage of LIKE Operator with Wildcard Characters

```
set serveroutput on;
declare
    v_name employees.first_name%type;
begin
    -- Fetch names starting with 'J'
    select first_name into v_name
    from employees
    where first_name like 'J%';

    dbms_output.put_line('Employee name: ' || v_name);

    -- Fetch names with '_' as a wildcard
    select first_name into v_name
    from employees
    where first_name like '_a%';

    dbms_output.put_line('Employee name with _ wildcard: ' || v_name);
end;
/
```

Output:

```
Employee name: [name_starting_with_J]
Employee name with _ wildcard: [name_with_a_as_second_character]
```

PROGRAM 6: Arrange Two Numbers in Ascending Order

```
set serveroutput on;
declare
    num1 number := &num1;
    num2 number := &num2;
    num_small number;
    num_large number;
begin
    if num1 < num2 then
        num_small := num1;
        num_large := num2;
    else
        num_small := num2;
        num_large := num1;
    end if;

    dbms_output.put_line('Small number: ' || num_small);
    dbms_output.put_line('Large number: ' || num_large);
end;
/
```

Output:

```
Small number: [smaller_number]
Large number: [larger_number]
```

PROGRAM 7: Procedure to Calculate Incentive on a Target Achieved

```
create or replace procedure calculate_incentive(p_emp_id
employees.employee_id%type, p_target number)
is
    v_incentive number(7,2);
    v_salary employees.salary%type;
begin
    select salary into v_salary
    from employees
    where employee_id = p_emp_id;

    if p_target >= 100000 then
        v_incentive := v_salary * 0.1;
        dbms_output.put_line('Incentive of ' || v_incentive || ' calculated
for employee ID ' || p_emp_id);
    else
        dbms_output.put_line('No incentive for employee ID ' || p_emp_id);
    end if;
end;
/
```

Output:

```
Incentive of [incentive_amount] calculated for employee ID [p_emp_id]
```

PROGRAM 8: Procedure to Calculate Incentive According to Specific Sale Limit

```
create or replace procedure calculate_incentive_on_sale(p_emp_id
employees.employee_id%type, p_sales number)
is
    v_incentive number(7,2);
begin
    if p_sales > 100000 then
        v_incentive := p_sales * 0.1;
    elsif p_sales between 50000 and 100000 then
        v_incentive := p_sales * 0.05;
    else
        v_incentive := 0;
    end if;

    dbms_output.put_line('Incentive for employee ID ' || p_emp_id || ' is:
' || v_incentive);
end;
/
```

Output:

```
Incentive for employee ID [p_emp_id] is: [incentive_amount]
```

PROGRAM 9: Count Employees in Department 50 and Check Vacancies

```
set serveroutput on;
declare
    v_emp_count number;
    v_vacancies number := 45;
begin
    select count(*) into v_emp_count
    from employees
    where department_id = 50;

    if v_emp_count < v_vacancies then
        dbms_output.put_line('Vacancies available: ' || (v_vacancies -
v_emp_count));
    else
        dbms_output.put_line('No vacancies available.');
    end if;
end;
/
```

Output:

```
Vacancies available: [number_of_vacancies]
```

PROGRAM 10: Count Employees in a Specific Department and Check Vacancies

```
set serveroutput on;
declare
    v_department_id number := &department_id;
    v_emp_count number;
    v_vacancies number := &vacancies;
begin
    select count(*) into v_emp_count
    from employees
    where department_id = v_department_id;

    if v_emp_count < v_vacancies then
        dbms_output.put_line('Vacancies available: ' || (v_vacancies -
v_emp_count));
    else
        dbms_output.put_line('No vacancies available.');
    end if;
end;
/
```

Output:

```
Vacancies available: [number_of_vacancies]
```

PROGRAM 11: Display Employee IDs, Names, Job Titles, Hire Dates, and Salaries

```
set serveroutput on;
begin
    for rec in (select employee_id, first_name || ' ' || last_name as name,
job_id, hire_date, salary from employees) loop
        dbms_output.put_line('ID: ' || rec.employee_id || ', Name: ' ||
rec.name || ', Job: ' || rec.job_id || ', Hire Date: ' || rec.hire_date ||
', Salary: ' || rec.salary);
    end loop;
end;
/
```

Output:

```
ID: [employee_id], Name: [name], Job: [job_id], Hire Date: [hire_date],
Salary: [salary]
```

PROGRAM 12: Display Employee IDs, Names, and Department Names

```
set serveroutput on;
begin
    for rec in (select e.employee_id, e.first_name || ' ' || e.last_name as
name, d.department_name
                from employees e
                join departments d on e.department_id = d.department_id)
loop
        dbms_output.put_line('ID: ' || rec.employee_id || ', Name: ' ||
rec.name || ', Department: ' || rec.department_name);
    end loop;
end;
/
```

Output:

```
ID: [employee_id], Name: [name], Department: [department_name]
```

PROGRAM 13: Display Job IDs, Titles, and Minimum Salaries

```
set serveroutput on;
begin
    for rec in (select job_id, job_title, min_salary from jobs) loop
        dbms_output.put_line('Job ID: ' || rec.job_id || ', Title: ' ||
rec.job_title || ', Min Salary: ' || rec.min_salary);
    end loop;
end;
/
```

Output:

```
Job ID: [job_id], Title: [job_title], Min Salary: [min_salary]
```

PROGRAM 14: Display Employee IDs, Names, and Job History Start Dates

```
set serveroutput on;
begin
    for rec in (select e.employee_id, e.first_name || ' ' || e.last_name as
name, j.start_date
                from employees e
                join job_history j on e.employee_id = j.employee_id) loop
        dbms_output.put_line('ID: ' || rec.employee_id || ', Name: ' ||
rec.name || ', Start Date: ' || rec.start_date);
    end loop;
end;
/
```

Output:

```
ID: [employee_id], Name: [name], Start Date: [start_date]
```

PROGRAM 15: Display Employee IDs, Names, and Job History End Dates

```
set serveroutput on;
begin
    for rec in (select e.employee_id, e.first_name || ' ' || e.last_name as
name, j.end_date
                from employees e
                join job_history j on e.employee_id = j.employee_id) loop
        dbms_output.put_line('ID: ' || rec.employee_id || ', Name: ' ||
rec.name || ', End Date: ' || rec.end_date);
    end loop;
end;
/
```

Output:

```
ID: [employee_id], Name: [name], End Date: [end_date]
```