

PL/SQL

NAME : VIDHYASHREE J

DEPARTMENT : AI&DS

ROLL NO: 241801310

EXERCISE : PL/SQL

PROGRAM 1

Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

-- 1. Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

-- 🎉 SOLUTION: Change 'employees' to 'HR.employees' 🎉

```
SET SERVEROUTPUT ON;
```

```
DECLARE  v_employee_id
NUMBER := 110;  v_salary
NUMBER;  v_incentive
NUMBER(10, 2);

BEGIN
    -- Select the salary into the variable
    SELECT salary
    INTO v_salary
    FROM HR.employees -- <<< FIX IS HERE
    WHERE employee_id = v_employee_id;
```

```
-- Calculate the incentive (10% of salary)
v_incentive := v_salary * 0.10;
```

```
-- Display the output
```

```
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id);  DBMS_OUTPUT.PUT_LINE('Salary:  
' || TO_CHAR(v_salary, 'FM999,999.00'));  DBMS_OUTPUT.PUT_LINE('Calculated Incentive: ' ||  
v_incentive);
```

EXCEPTION

```
WHEN NO_DATA_FOUND THEN  
    DBMS_OUTPUT.PUT_LINE('Employee with ID ' || v_employee_id || ' not found.');//  
WHEN OTHERS THEN  
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);  
END;
```

```
/
```

Query result **Script output** DBMS output Explain Plan SQL history



```
Employee ID: 110  
Salary: 8,200.00  
Calculated Incentive: 820
```

```
PL/SQL procedure successfully completed.
```

```
Elapsed: 00:00:00.005
```

PROGRAM 2

Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

```
DECLARE
```

```
    "MyVar" NUMBER := 10;
```

```
myvar NUMBER := 20;
```

```
BEGIN
```

```
    -- Unquoted references are case-insensitive (defaults to MYVAR = 20)
```

```
    DBMS_OUTPUT.PUT_LINE('Reference 1 (Unquoted variable): ' || MyVar);
```

```
    -- Quoted references must use the exact case (refers to "MyVar" = 10)
```

```
    DBMS_OUTPUT.PUT_LINE('Reference 2 (Quoted variable): ' || "MyVar");
```

```
END;
```

Query result Script output **DBMS output** Explain Plan SQL history



BEGIN...

Show more...

```
Reference 1 (Unquoted variable): 20  
Reference 2 (Quoted variable): 10
```

```
PL/SQL procedure successfully completed.
```

```
Elapsed: 00:00:00.005
```

[Terms and Conditions](#) | [Your Privacy Rights](#) | [Cookie Preferences](#)

r40 - Copyright © 2015, 2025 Oracle and/or its affiliates. All rights reserved.

PROGRAM 3

Write a PL/SQL block to adjust the salary of the employee whose ID 122.

Sample table: employees

```
SET SERVEROUTPUT ON;
```

```
DECLARE
```

```
    -- Use your local table name  v_old_salary
```

```
    MY_EMPLOYEES.salary%TYPE;  v_new_salary
```

```
    MY_EMPLOYEES.salary%TYPE;
```

```
    v_employee_id CONSTANT NUMBER := 122;
```

```
BEGIN
```

```
    -- Use your local table name
```

```
    SELECT salary
```

```
        INTO v_old_salary
```

```
        FROM MY_EMPLOYEES
```

```
        WHERE employee_id = v_employee_id;
```

```
    v_new_salary := v_old_salary * 1.10; -- 10% increase
```

```
    -- Use your local table name
```

```
    UPDATE MY_EMPLOYEES
```

```

SET salary = v_new_salary
WHERE employee_id = v_employee_id;

COMMIT; -- Save the changes

DBMS_OUTPUT.PUT_LINE('Salary adjusted successfully.');
DBMS_OUTPUT.PUT_LINE('Old Salary: ' || v_old_salary);
DBMS_OUTPUT.PUT_LINE('New Salary: ' || v_new_salary);

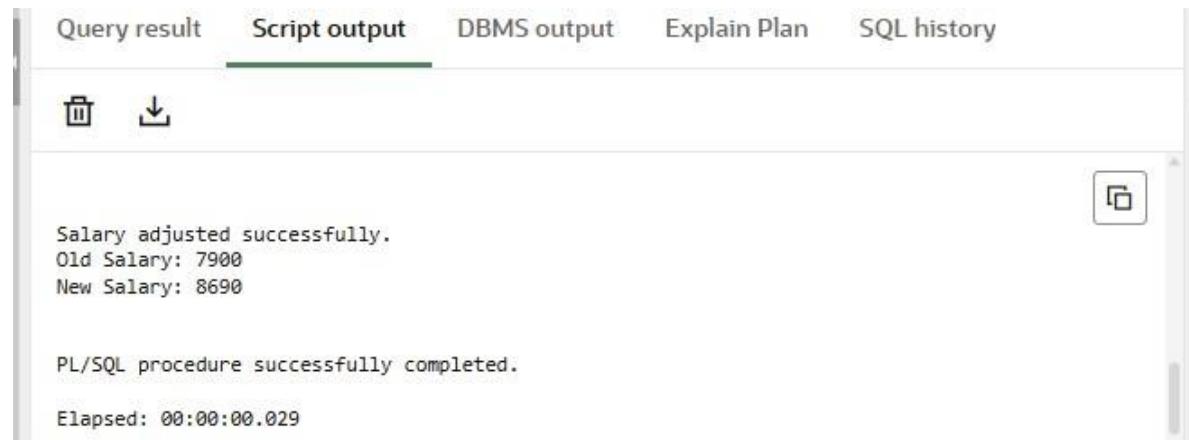
```

EXCEPTION

```

WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Employee with ID ' || v_employee_id || ' not found.');
/

```



The screenshot shows the Oracle SQL Developer interface with the 'Script output' tab selected. The output window displays the following text:

```

Salary adjusted successfully.
Old Salary: 7900
New Salary: 8690

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.029

```

PROGRAM 4

Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

```
SET SERVEROUTPUT ON;
```

```

-- 1. Create the Procedure
-- This procedure demonstrates that the AND condition fails if one operand is FALSE
-- (which is the result of any comparison involving NULL, like IS NOT NULL, when the variable IS
NULL).

```

```

CREATE OR REPLACE PROCEDURE check_null_and_condition IS
    v_num1 NUMBER := 10;    v_num2 NUMBER := NULL; -- The
    key variable set to NULL

BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Testing AND Operator with NULL ---');

    -- Condition 1: v_num1 IS NOT NULL (TRUE) AND v_num1 > 5 (TRUE) -> TRUE
    IF v_num1 IS NOT NULL AND v_num1 > 5 THEN
        DBMS_OUTPUT.PUT_LINE('Condition 1: TRUE (10 is NOT NULL AND 10 > 5)');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Condition 1: FALSE');
    END IF;

    -- Condition 2: v_num2 IS NOT NULL (FALSE) AND v_num1 > 5 (TRUE) -> FALSE
    -- The AND operator requires BOTH sides to be TRUE. Since the first side
    -- (NULL IS NOT NULL) is FALSE, the entire condition fails immediately.
    IF v_num2 IS NOT NULL AND v_num1 > 5 THEN
        DBMS_OUTPUT.PUT_LINE('Condition 2: TRUE');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Condition 2: FALSE (v_num2 is NULL, so the condition fails)');
    END IF;
END;
/
-- 2. Execute the Procedure BEGIN
check_null_and_condition;
END;
/

```

The screenshot shows a software interface with a top navigation bar containing tabs: 'Query result', 'Script output' (which is highlighted in green), 'DBMS output', 'Explain Plan', and 'SQL history'. Below the tabs is a toolbar with icons for delete, save, and copy. The main area contains the following text:

```
--- Testing AND Operator with NULL ---
Condition 1: TRUE (10 is NOT NULL AND 10 > 5)
Condition 2: FALSE (v_num2 is NULL, so the condition fails)

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.004
```

PROGRAM 5

Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

```
SET SERVEROUTPUT ON;
```

```
DECLARE  v_name VARCHAR2(50) :=
'JOHN_SMITH';

BEGIN
    -- 1. Using the % wildcard (Matches prefix)
    IF v_name LIKE 'JOHN%' THEN
        DBMS_OUTPUT.PUT_LINE('1. Matches pattern: JOHN% (Matches strings starting with JOHN)');
    END IF;

    -- 2. Using the % wildcard (Matches suffix)
    IF v_name LIKE '%SMITH' THEN
        DBMS_OUTPUT.PUT_LINE('2. Matches pattern: %SMITH (Matches strings ending with SMITH)');
    END IF;

    -- 3. Using the _ wildcard (Matches any single character)
    -- Note: The underscore in 'JOHN_SMITH' is a literal character here.
    IF v_name LIKE 'J_HN%' THEN
        DBMS_OUTPUT.PUT_LINE('3. Matches pattern: J_HN% (The _ matches the O in JOHN_SMITH)');
    END IF;
```

```
-- 4. ESCAPE Clause (Syntax only example in original code)

IF v_name LIKE 'J_HN%' ESCAPE '\' THEN

    DBMS_OUTPUT.PUT_LINE('4. ESCAPE Clause is demonstrated (Syntax is correct but not strictly
required for this specific match)");

END IF;

END;
/

```

The screenshot shows the Oracle SQL Developer interface with the 'Script output' tab selected. The results pane displays the following output:

```

1. Matches pattern: JOHN% (Matches strings starting with JOHN)
2. Matches pattern: %SMITH (Matches strings ending with SMITH)
3. Matches pattern: J_HN% (The _ matches the 0 in JOHN_SMITH)
4. ESCAPE Clause is demonstrated (Syntax is correct but not strictly required for this specific

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.002

```

PROGRAM 6

Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable.

```
SET SERVEROUTPUT ON;
```

```

DECLARE  num1
NUMBER:=45;  num2
NUMBER:=20;
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: '||num_small);
DBMS_OUTPUT.PUT_LINE('Large: '||num_large);
END;
/

```

Query result Script output DBMS output Explain Plan SQL history

Show more...

```

Small: 20
Large: 45

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.006

```

PROGRAM 7

Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

SET SERVEROUTPUT ON;

-- 1. Procedure Creation: Run this block first to define the procedure.

```

CREATE OR REPLACE PROCEDURE calc_incentive(p_emp_id NUMBER,p_target NUMBER) IS
v_incentive NUMBER;
BEGIN
IF p_target>=10000 THEN
v_incentive:=p_target*0.1;
DBMS_OUTPUT.PUT_LINE('Incentive: '||v_incentive);

```

```

-- The program logic specifies 'Record updated', though no UPDATE is performed.

DBMS_OUTPUT.PUT_LINE('Record updated');

ELSE

    DBMS_OUTPUT.PUT_LINE('No incentive, record not updated');

END IF;

END;

/

```

-- 2. Execution Block: Run this block second to call the procedure.

```

BEGIN calc_incentive(101,12000); -- Employee 101 with a target
of 12000

END;
/

```

The screenshot shows a SQL developer interface with the following details:

- Query result**: Shows the output of the executed PL/SQL block.
- Script output**: The active tab, showing the executed code:


```
SQL> BEGIN
      calc_incentive(101,12000); -- Employee 101 with a target of 12000
      END;
```
- DBMS output**: Shows the results of the DBMS_OUTPUT.PUT_LINE statements:


```
Incentive: 1200
Record updated
```
- Explain Plan**: Not selected.
- SQL history**: Not selected.

At the bottom, there are links for Terms and Conditions, Your Privacy Rights, Delete Your FreeSQL Account, and Cookie Preferences.

PROGRAM 8

Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.

```
SET SERVEROUTPUT ON;
```

-- 1. Procedure Creation: Run this block first to define the procedure.

```

CREATE OR REPLACE PROCEDURE incentive_by_sale(p_emp_id NUMBER,p_sales NUMBER) IS
    v_incentive NUMBER:=0;
BEGIN

```

```

IF p_sales >= 20000 THEN      v_incentive := p_sales *
0.15; -- 15% for sales >= 20,000  ELSIF p_sales >= 10000
THEN      v_incentive := p_sales * 0.10; -- 10% for sales >=
10,000  ELSIF p_sales >= 5000 THEN      v_incentive :=
p_sales * 0.05; -- 5% for sales >= 5,000  ELSE
v_incentive := 0; -- 0% for sales < 5,000
END IF;

```

```

DBMS_OUTPUT.PUT_LINE('Incentive: '||v_incentive);
END;
/

```

-- 2. Execution Block: Run this block second to call the procedure.

```

BEGIN  incentive_by_sale(101,15000); -- Employee 101 with sales of
15,000
END;
/

```



The screenshot shows the Oracle SQL Developer interface with the 'Script output' tab selected. The output window displays the following:

```

Query result Script output DBMS output Explain Plan SQL history

SQL> BEGIN
      incentive_by_sale(101,15000); -- Employee 101 with sales of 15,000
    END;

Incentive: 1500

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.004

```

PROGRAM 9

Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

```
SET SERVEROUTPUT ON;
```

```

DECLARE  v_count
NUMBER;  v_vac
NUMBER;

-- Set the total number of vacancies based on the problem description (45)

v_total_vacancies CONSTANT NUMBER := 45;

BEGIN

-- Count employees in department 50

SELECT COUNT(*) INTO v_count

FROM HR.employees -- FIX: Schema-qualified the table name

WHERE department_id=50;

-- Calculate vacancies  v_vac :=

v_total_vacancies - v_count;

IF v_vac > 0 THEN

DBMS_OUTPUT.PUT_LINE('Vacancies available: '||v_vac);

ELSE

DBMS_OUTPUT.PUT_LINE('No vacancies');

END IF;

END;
/

```



The screenshot shows the Oracle SQL Developer interface with the 'Script output' tab selected. The window displays the following content:

- Query result**, **Script output** (selected), **DBMS output**, **Explain Plan**, **SQL history**
- Icons for **trash** and **download**.
- SQL command: `SQL> DECLARE`
- Procedure code (continued from the previous block):
`v_count NUMBER;
v_vac NUMBER;
-- Set the total number of vacancies based on the problem description (45)...
Show more...`
- Output:
`No vacancies`
- Message at the bottom:
`PL/SQL procedure successfully completed.`

PROGRAM 10

Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.

```
SET SERVEROUTPUT ON;
```

```
DECLARE
```

```
    v_count NUMBER;    v_vac NUMBER;    v_dept  
    NUMBER:=60; -- Department ID to check    v_total  
    NUMBER:=45; -- Total available positions/vacancies
```

```
BEGIN
```

```
    -- Count employees in the specified department  
    SELECT COUNT(*) INTO v_count  
    FROM HR.employees -- FIX: Schema-qualified the table name  
    WHERE department_id=v_dept;
```

```
    -- Calculate vacancies
```

```
v_vac := v_total - v_count;
```

```
IF v_vac > 0 THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Department'||v_dept||' has '||v_vac||' vacancies');
```

```
ELSE
```

```
    DBMS_OUTPUT.PUT_LINE('No vacancies in department '||v_dept);
```

```
END IF;
```

```
END;
```

```
/
```

The screenshot shows the Oracle SQL Developer interface with the 'Script output' tab selected. The output window displays the following PL/SQL code and its execution results:

```
SQL> DECLARE
  V_count NUMBER;
  V_vac NUMBER;
  V_dept NUMBER:=60;    -- Department ID to check...
Show more...

Department 60 has 40 vacancies

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.009
```

Below the output window, there is a navigation bar with links: About Oracle | Contact Us | Legal Notices | Terms and Conditions | Your Privacy Rights | Delete Your FreeSQL Account.

PROGRAM 11

Write a PL/SQL program to display the employee IDs, names, job titles, hire dates, and salaries of all employees.

```
SET SERVEROUTPUT ON;
```

```
BEGIN
```

```
-- FIX: Schema-qualify the table name
```

```
FOR r IN (SELECT employee_id, first_name ||' '| last_name AS name, job_id, hire_date, salary
          FROM HR.employees)
```

```
LOOP
```

```
    DBMS_OUTPUT.PUT_LINE(r.employee_id ||''|| r.name ||''|| r.job_id ||''|| r.hire_date ||'||'
                           r.salary);
```

```
END LOOP;
```

```
END;
```

```
/
```

[Query result](#) [Script output](#) [DBMS output](#) [Explain Plan](#) [SQL history](#)



```
SQL> BEGIN
  -- FIX: Schema-qualify the table name
  FOR r IN (SELECT employee_id, first_name ||' '| last_name AS name, job_id, hire_date, salary
            FROM HR.employees)...
  Show more...
```

```
199 Douglas Grant SH_CLERK 13-JAN-18 2600
200 Jennifer Whalen AD_ASST 17-SEP-13 4400
201 Michael Martinez MK_MAN 17-FEB-14 13000
202 Pat Davis MK_REP 17-AUG-15 6000
203 Susan Jacobs HR_REP 07-JUN-12 6500
204 Hermann Brown PR_REP 07-JUN-12 10000
205 Shelley Higgins AC_MGR 07-JUN-12 12008
206 William Gietz AC_ACCOUNT 07-JUN-12 8300
100 Steven King AD_PRES 17-JUN-13 24000
101 Neena Yang AD_VP 21-SEP-15 17000
...  
102 Lex Garcia AD_VP 13-JAN-11 17000
103 Alexander James IT_PROG 03-JAN-16 9000
104 Bruce Miller IT_PROG 21-MAY-17 6000
105 David Williams IT_PROG 25-JUN-15 4800
106 Valli Jackson IT_PROG 05-FEB-16 4200
107 Diana Nguyen IT_PROG 07-FEB-17 4200
108 Nancy Gruenberg FI_MGR 17-AUG-12 12008
109 Daniel Faviet FI_ACCOUNT 16-AUG-12 9000
110 John Chen FI_ACCOUNT 28-SEP-15 8200
111 Ismael Sciarra FI_ACCOUNT 30-SEP-15 7700
112 Jose Manuel Urman FI_ACCOUNT 07-MAR-16 7800
113 Luis Popp FI_ACCOUNT 07-DEC-17 6900
114 Den Li PU_MAN 07-DEC-12 11000
115 Alexander Khoo PU_CLERK 18-MAY-13 3100
116 Shelli Baida PU_CLERK 24-DEC-15 2900
117 Sigal Tobias PU_CLERK 24-JUL-15 2800
118 Guy Imuro PU_CLERK 15-NOV-16 2600
119 Karen Colmenares PU_CLERK 10-AUG-17 2500
...  
120 Matthew Weiss ST_MAN 18-JUL-14 8000
121 Adam Fripp ST_MAN 10-APR-15 8200
122 Payam Kaufling ST_MAN 01-MAY-13 7900
123 Shanta Vollman ST_MAN 10-OCT-15 6500
124 Kevin Mourgos ST_MAN 16-NOV-17 5800
125 Julia Nayer ST_CLERK 16-JUL-15 3200
126 Irene Mikkilineni ST_CLERK 28-SEP-16 2700
127 James Landry ST_CLERK 14-JAN-17 2400
128 Steven Markle ST_CLERK 08-MAR-18 2200
129 Laura Bissot ST_CLERK 20-AUG-15 3300
130 Mozhe Atkinson ST_CLERK 30-OCT-15 2800
131 James Marlow ST_CLERK 16-FEB-15 2500
132 TJ Olson ST_CLERK 10-APR-17 2100
133 Jason Mallin ST_CLERK 14-JUN-14 3300
134 Michael Rogers ST_CLERK 26-AUG-16 2900
135 Ki Gee ST_CLERK 12-DEC-17 2400
136 Hazel Philtanker ST_CLERK 06-FEB-18 2200
137 Renske Ladwig ST_CLERK 14-JUL-13 3600
138 Stephan Stiles ST_CLERK 26-OCT-15 2200
```

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your FreeSQL Account](#) | [Cookie Preferences](#)

[Query result](#) [Script output](#) [DBMS output](#) [Explain Plan](#) [SQL history](#)



```
102 Lex Garcia AD_VP 13-JAN-11 17000
103 Alexander James IT_PROG 03-JAN-16 9000
104 Bruce Miller IT_PROG 21-MAY-17 6000
105 David Williams IT_PROG 25-JUN-15 4800
106 Valli Jackson IT_PROG 05-FEB-16 4200
107 Diana Nguyen IT_PROG 07-FEB-17 4200
108 Nancy Gruenberg FI_MGR 17-AUG-12 12008
109 Daniel Faviet FI_ACCOUNT 16-AUG-12 9000
110 John Chen FI_ACCOUNT 28-SEP-15 8200
111 Ismael Sciarra FI_ACCOUNT 30-SEP-15 7700
112 Jose Manuel Urman FI_ACCOUNT 07-MAR-16 7800
113 Luis Popp FI_ACCOUNT 07-DEC-17 6900
114 Den Li PU_MAN 07-DEC-12 11000
115 Alexander Khoo PU_CLERK 18-MAY-13 3100
116 Shelli Baida PU_CLERK 24-DEC-15 2900
117 Sigal Tobias PU_CLERK 24-JUL-15 2800
118 Guy Imuro PU_CLERK 15-NOV-16 2600
119 Karen Colmenares PU_CLERK 10-AUG-17 2500
...  
120 Matthew Weiss ST_MAN 18-JUL-14 8000
121 Adam Fripp ST_MAN 10-APR-15 8200
122 Payam Kaufling ST_MAN 01-MAY-13 7900
123 Shanta Vollman ST_MAN 10-OCT-15 6500
124 Kevin Mourgos ST_MAN 16-NOV-17 5800
125 Julia Nayer ST_CLERK 16-JUL-15 3200
126 Irene Mikkilineni ST_CLERK 28-SEP-16 2700
127 James Landry ST_CLERK 14-JAN-17 2400
128 Steven Markle ST_CLERK 08-MAR-18 2200
129 Laura Bissot ST_CLERK 20-AUG-15 3300
130 Mozhe Atkinson ST_CLERK 30-OCT-15 2800
131 James Marlow ST_CLERK 16-FEB-15 2500
132 TJ Olson ST_CLERK 10-APR-17 2100
133 Jason Mallin ST_CLERK 14-JUN-14 3300
134 Michael Rogers ST_CLERK 26-AUG-16 2900
135 Ki Gee ST_CLERK 12-DEC-17 2400
136 Hazel Philtanker ST_CLERK 06-FEB-18 2200
137 Renske Ladwig ST_CLERK 14-JUL-13 3600
138 Stephan Stiles ST_CLERK 26-OCT-15 2200
```

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your FreeSQL Account](#) | [Cookie Preferences](#)

[Query result](#) [Script output](#) [DBMS output](#) [Explain Plan](#) [SQL history](#)



```
120 Matthew Weiss ST_MAN 18-JUL-14 8000
121 Adam Fripp ST_MAN 10-APR-15 8200
122 Payam Kaufling ST_MAN 01-MAY-13 7900
123 Shanta Vollman ST_MAN 10-OCT-15 6500
124 Kevin Mourgos ST_MAN 16-NOV-17 5800
125 Julia Nayer ST_CLERK 16-JUL-15 3200
126 Irene Mikkilineni ST_CLERK 28-SEP-16 2700
127 James Landry ST_CLERK 14-JAN-17 2400
128 Steven Markle ST_CLERK 08-MAR-18 2200
129 Laura Bissot ST_CLERK 20-AUG-15 3300
130 Mozhe Atkinson ST_CLERK 30-OCT-15 2800
131 James Marlow ST_CLERK 16-FEB-15 2500
132 TJ Olson ST_CLERK 10-APR-17 2100
133 Jason Mallin ST_CLERK 14-JUN-14 3300
134 Michael Rogers ST_CLERK 26-AUG-16 2900
135 Ki Gee ST_CLERK 12-DEC-17 2400
136 Hazel Philtanker ST_CLERK 06-FEB-18 2200
137 Renske Ladwig ST_CLERK 14-JUL-13 3600
138 Stephan Stiles ST_CLERK 26-OCT-15 2200
```

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your FreeSQL Account](#) | [Cookie Preferences](#)

[Query result](#) [Script output](#) [DBMS output](#) [Explain Plan](#) [SQL history](#)



139 John Seo ST_CLERK 12-FEB-16 2700
140 Joshua Patel ST_CLERK 06-APR-16 2500
141 Trenna Rajts ST_CLERK 17-OCT-13 3500
142 Curtis Davies ST_CLERK 29-JAN-15 3100
143 Randall Matos ST_CLERK 15-MAR-16 2600
144 Peter Vargas ST_CLERK 09-JUL-16 2500
145 John Singh SA_MAN 01-OCT-14 14000
146 Karen Partners SA_MAN 05-JAN-15 13500
147 Alberto Errazuriz SA_MAN 10-MAR-15 12000
148 Gerald Cambrault SA_MAN 15-OCT-17 11000
149 Eleni Zlotkey SA_MAN 29-JAN-16 10500
150 Sean Tucker SA REP 30-JAN-15 10000
151 David Bernstein SA REP 24-MAR-15 9500
152 Peter Hall SA REP 20-AUG-15 9000
153 Christopher Olsen SA REP 30-MAR-16 8000
154 Nanette Cambrault SA REP 09-DEC-16 7500
155 Oliver Tuvauit SA REP 23-NOV-17 7000
156 Janette King SA REP 30-JAN-14 10000
157 Patrick Sully SA RFP 04-MAR-14 9500

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your FreeSQL Account](#) | [Cookie Preferences](#)

[Query result](#) [Script output](#) [DBMS output](#) [Explain Plan](#) [SQL history](#)



160 Louise Doran SA REP 15-DEC-15 7500
161 Sarah Sewall SA REP 03-NOV-16 7000
162 Clara Vishney SA REP 11-NOV-15 10500
163 Danielle Greene SA REP 19-MAR-17 9500
164 Mattea Marvins SA REP 24-JAN-18 7200
165 David Lee SA REP 23-FEB-18 6800
166 Sundar Ande SA REP 24-MAR-18 6400
167 Amit Banda SA REP 21-APR-18 6200
168 Lisa Ozer SA REP 11-MAR-15 11500
169 Harrison Bloom SA REP 23-MAR-16 10000
170 Tayler Fox SA REP 24-JAN-16 9600
171 William Smith SA REP 23-FEB-17 7400
172 Elizabeth Bates SA REP 24-MAR-17 7300
173 Sundita Kumar SA REP 21-APR-18 6100
174 Ellen Abel SA REP 11-MAY-14 11000
175 Alyssa Hutton SA REP 19-MAR-15 8800
176 Jonathon Taylor SA REP 24-MAR-16 8600
177 Jack Livingston SA REP 23-APR-16 8400
178 Kimherelv Grant SA RFP 24-MAY-17 7000

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your FreeSQL Account](#) | [Cookie Preferences](#)

[Query result](#) [Script output](#) [DBMS output](#) [Explain Plan](#) [SQL history](#)



186 Julia Dellinger SH_CLERK 24-JUN-16 3400
187 Anthony Cabrio SH_CLERK 07-FEB-17 3000
188 Kelly Chung SH_CLERK 14-JUN-15 3800
189 Jennifer Dilly SH_CLERK 13-AUG-15 3600
190 Timothy Venzl SH_CLERK 11-JUL-16 2900
191 Randall Perkins SH_CLERK 19-DEC-17 2500
192 Sarah Bell SH_CLERK 04-FEB-14 4000
193 Britney Everett SH_CLERK 03-MAR-15 3900
194 Samuel McLeod SH_CLERK 01-JUL-16 3200
195 Vance Jones SH_CLERK 17-MAR-17 2800
196 Alana Walsh SH_CLERK 24-APR-16 3100
197 Kevin Feeney SH_CLERK 23-MAY-16 3000
198 Donald O'Connell SH_CLERK 21-JUN-17 2600

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.012

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your FreeSQL Account](#) | [Cookie Preferences](#)

PROGRAM 12

Write a PL/SQL program to display the employee IDs, names, and department names of all employees.

```
SET SERVEROUTPUT ON;
```

```
BEGIN
```

```
-- FIX: Schema-qualify the tables to HR.employees and HR.departments  
FOR r IN (SELECT e.employee_id, e.first_name||' '||e.last_name AS name, d.department_name  
          FROM HR.employees e JOIN HR.departments d ON e.department_id=d.department_id)
```

```
LOOP
```

```
    DBMS_OUTPUT.PUT_LINE(r.employee_id ||' '|| r.name ||' '|| r.department_name);
```

```
END LOOP;
```

```
END;
```

```
/
```

The screenshot shows a SQL developer interface with the 'Script output' tab selected. The code is displayed in a scrollable window, and the results of the execution are shown below it. The results list employee details such as ID, first name, last name, and department name.

```
SQL> BEGIN  
  -- FIX: Schema-qualify the tables to HR.employees and HR.departments  
  FOR r IN (SELECT e.employee_id, e.first_name||' '||e.last_name AS name, d.department_name  
            FROM HR.employees e JOIN HR.departments d ON e.department_id=d.department_id)...  
Show more...  
  
200 Jennifer Whalen Administration  
201 Michael Martinez Marketing  
202 Pat Davis Marketing  
114 Den Li Purchasing  
119 Karen Colmenares Purchasing  
115 Alexander Khoo Purchasing  
116 Shelli Baida Purchasing  
117 Sigal Tobias Purchasing  
118 Guy Himuro Purchasing  
203 Susan Jacobs Human Resources
```

Query result **Script output** DBMS output Explain Plan SQL history



```
122 Payam Kaufling Shipping
123 Shanta Vollman Shipping
124 Kevin Mourgos Shipping
125 Julia Nayer Shipping
126 Irene Mikkilineni Shipping
127 James Landry Shipping
128 Steven Markle Shipping
129 Laura Bissot Shipping
130 Mozhe Atkinson Shipping
131 James Marlow Shipping
132 TJ Olson Shipping
133 Jason Mallin Shipping
134 Michael Rogers Shipping
135 Ki Gee Shipping
136 Hazel Philtanker Shipping
137 Renske Ladwig Shipping
138 Stephen Stiles Shipping
139 John Scaachi Shipping
```

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your FreeSQL Account](#)

Query result **Script output** DBMS output Explain Plan SQL history



```
182 Marlene Sullivan Shipping
183 Girard Geoni Shipping
184 Nandita Sarchand Shipping
185 Alexis Bull Shipping
186 Julia Dellinger Shipping
187 Anthony Cabrio Shipping
188 Kelly Chung Shipping
189 Jennifer Dilly Shipping
190 Timothy Venzl Shipping
191 Randall Perkins Shipping
192 Sarah Bell Shipping
193 Britney Everett Shipping
194 Samuel McLeod Shipping
195 Vance Jones Shipping
196 Alana Walsh Shipping
197 Kevin Feeney Shipping
198 Donald O'Connell Shipping
199 Bruce Miller IT
200 Alphonse Greene IT
```

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your FreeSQL Account](#)

Query result **Script output** DBMS output Explain Plan SQL history



```
167 Amit Banda Sales
101 Neena Yang Executive
100 Steven King Executive
102 Lex Garcia Executive
110 John Chen Finance
108 Nancy Gruenberg Finance
111 Ismael Sciarra Finance
112 Jose Manuel Urman Finance
113 Luis Popp Finance
109 Daniel Faviet Finance
206 William Gietz Accounting
205 Shelley Higgins Accounting
```

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.012

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your FreeSQL Account](#) | [Cookies](#)

PROGRAM 13

Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.

```
SET SERVEROUTPUT ON;
```

```
BEGIN
```

```
-- FIX: Schema-qualify the table to HR.jobs
```

```
FOR r IN (SELECT job_id, job_title, min_salary
```

```
      FROM HR.jobs)
```

```
LOOP
```

```
-- FIX: Corrected concatenation from ' || ' to ' || '
```

```
    DBMS_OUTPUT.PUT_LINE(r.job_id || '||' || r.job_title || '||' || r.min_salary);
```

```
END LOOP;
```

```
END;
```

```
/
```

Query result **Script output** DBMS output Explain Plan SQL history

SQL> BEGIN
-- FIX: Schema-qualify the table to HR.jobs
FOR r IN (SELECT job_id, job_title, min_salary
FROM HR.jobs)...
Show more...

```
AD_PRES President 20080
AD_VP Administration Vice President 15000
AD_ASST Administration Assistant 3000
FI_MGR Finance Manager 8200
FI_ACCOUNT Accountant 4200
AC_MGR Accounting Manager 8200
AC_ACCOUNT Public Accountant 4200
SA_MAN Sales Manager 10000
SA_REP Sales Representative 6000
PU_MAN Purchasing Manager 8000
```

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your FreeSQL Account](#) | [Cookie Preference](#)

Query result **Script output** DBMS output Explain Plan SQL history

SQL> BEGIN
SA_MAN Sales Manager 10000
SA_REP Sales Representative 6000
PU_MAN Purchasing Manager 8000
PU_CLERK Purchasing Clerk 2500
ST_MAN Stock Manager 5500
ST_CLERK Stock Clerk 2000
SH_CLERK Shipping Clerk 2500
IT_PROG Programmer 4000
MK_MAN Marketing Manager 9000
MK_REP Marketing Representative 4000
HR_REP Human Resources Representative 4000
PR_REP Public Relations Representative 4500

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.009

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your FreeSQL Account](#) | [Cookie Preference](#)

PROGRAM 14

Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

SET SERVEROUTPUT ON;

BEGIN

-- FIX: Schema-qualify the tables

```
FOR r IN (SELECT e.employee_id, e.first_name||' '||e.last_name AS name, jh.start_date
FROM HR.employees e JOIN HR.job_history jh ON e.employee_id=jh.employee_id)
```

LOOP

```

DBMS_OUTPUT.PUT_LINE(r.employee_id ||' '|| r.name ||' '|| r.start_date);

END LOOP;

END;
/

```

Query result Script output DBMS output Explain Plan SQL history

```

SQL> BEGIN
  -- FIX: Schema-qualify the tables
  FOR r IN (SELECT e.employee_id, e.first_name||' '||e.last_name AS name, jh.start_date
            FROM HR.employees e JOIN HR.job_history jh ON e.employee_id=jh.employee_id)...
  Show more...

```

```

101 Neena Yang 21-SEP-07
101 Neena Yang 28-OCT-11
102 Lex Garcia 13-JAN-11
114 Den Li 24-MAR-16
122 Payam Kaufling 01-JAN-17
176 Jonathon Taylor 24-MAR-16
176 Jonathon Taylor 01-JAN-17
200 Jennifer Whalen 17-SEP-05
200 Jennifer Whalen 01-JUL-12
201 Michael Martinez 17-FEB-14

```

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your FreeSQL Account](#) | [Cookie Preferences](#)

Query result Script output DBMS output Explain Plan SQL history

```

101 Neena Yang 21-SEP-07
101 Neena Yang 28-OCT-11
102 Lex Garcia 13-JAN-11
114 Den Li 24-MAR-16
122 Payam Kaufling 01-JAN-17
176 Jonathon Taylor 24-MAR-16
176 Jonathon Taylor 01-JAN-17
200 Jennifer Whalen 17-SEP-05
200 Jennifer Whalen 01-JUL-12
201 Michael Martinez 17-FEB-14

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.012

```

[About Oracle](#) | [Contact Us](#) | [Legal Notices](#) | [Terms and Conditions](#) | [Your Privacy Rights](#) | [Delete Your FreeSQL Account](#) | [Cookie Preferences](#)

PROGRAM 15

Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

Query result **Script output** DBMS output Explain Plan SQL history



```
101 Neena Yang 27-OCT-11  
101 Neena Yang 15-MAR-15  
102 Lex Garcia 24-JUL-16  
114 Den Li 31-DEC-17  
122 Payam Kaufling 31-DEC-17  
176 Jonathon Taylor 31-DEC-16  
176 Jonathon Taylor 31-DEC-17  
200 Jennifer Whalen 17-JUN-11  
200 Jennifer Whalen 31-DEC-16  
201 Michael Martinez 19-DEC-17
```

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.011