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;
/
```

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;

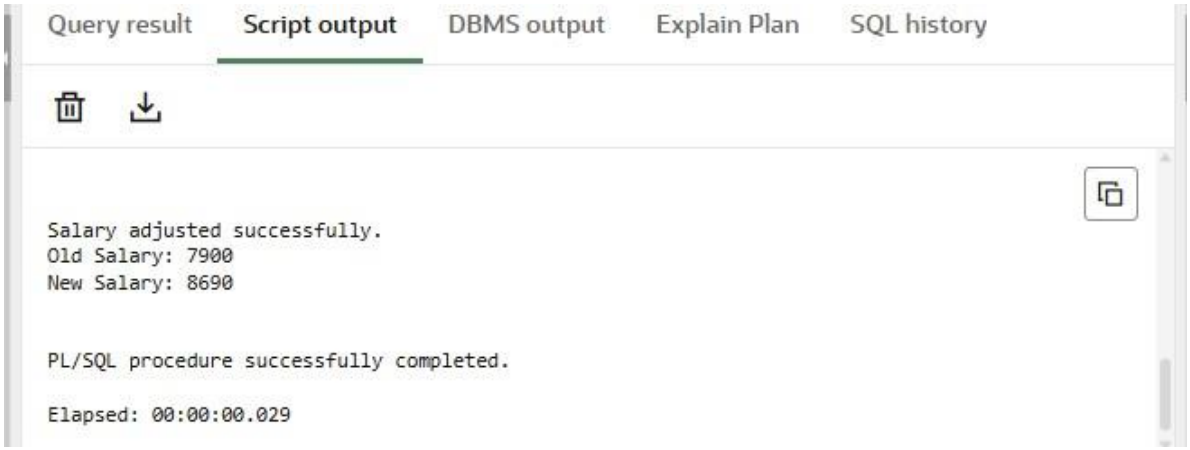
PROGRAM 3

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.'); END;
/
```



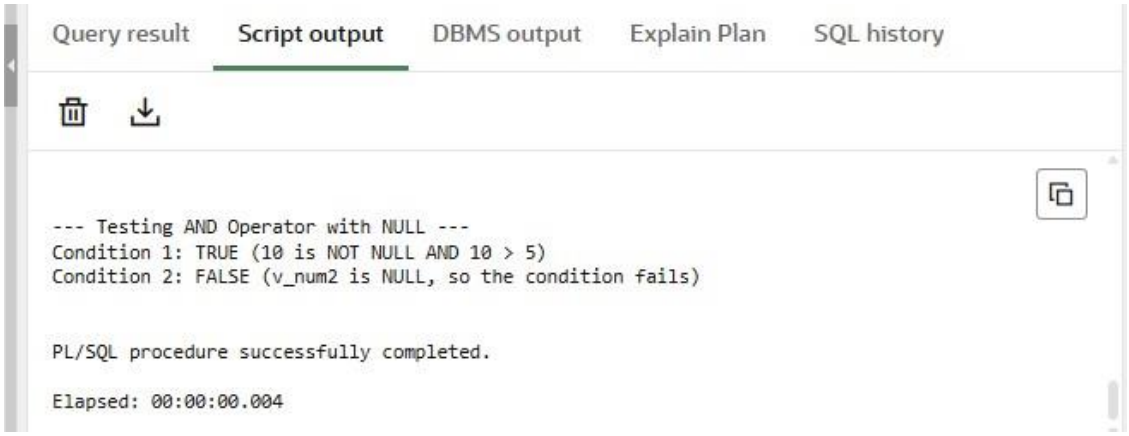
PROGRAM 4

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;
/
```

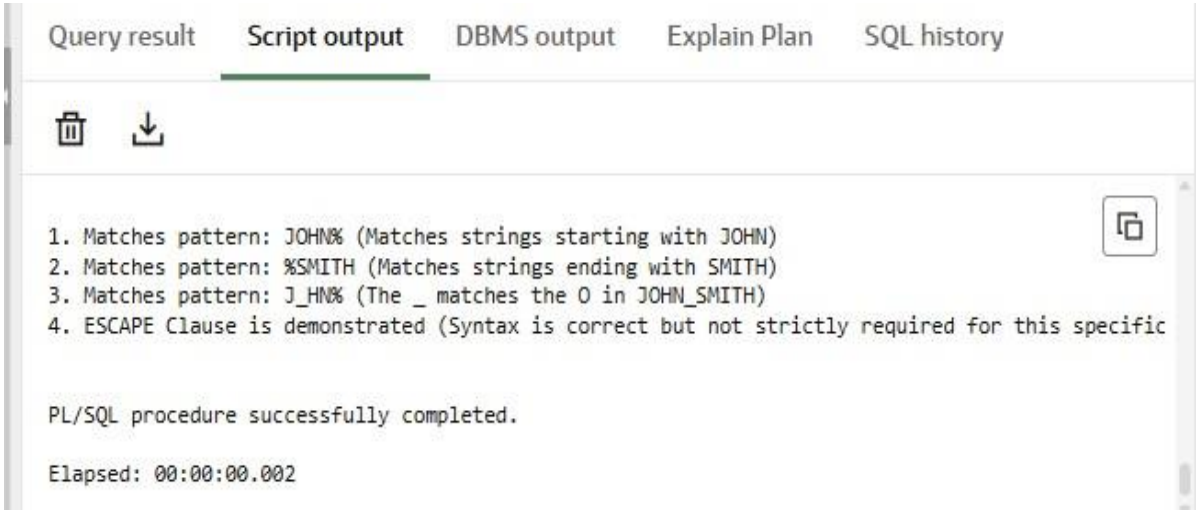


PROGRAM 5

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;
/
```



PROGRAM 6

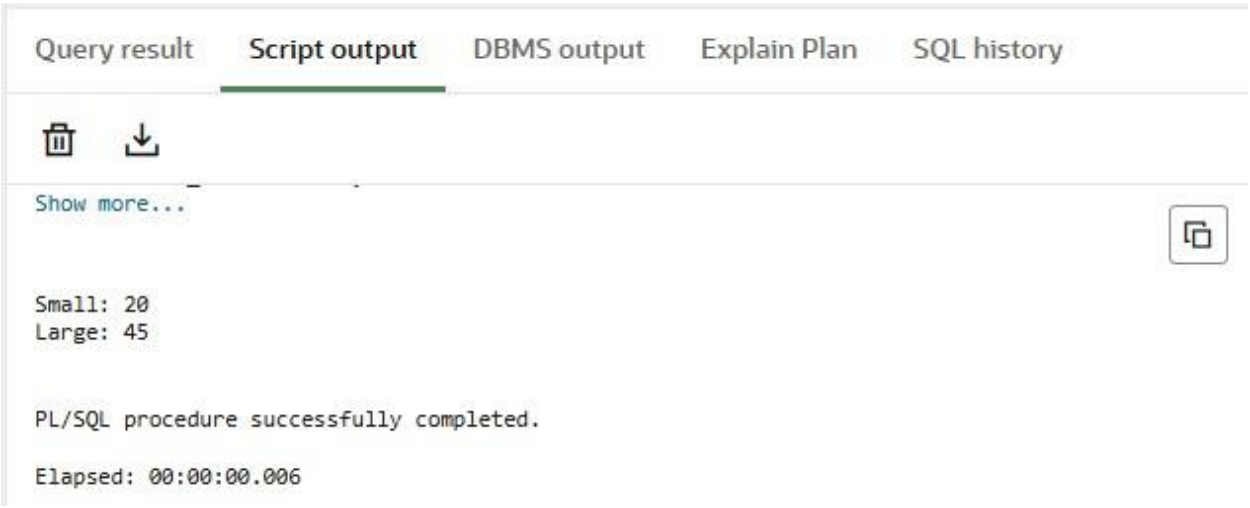
Write a PL/SQL program to arrange the number of two variable in such a way that the

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;
/
```

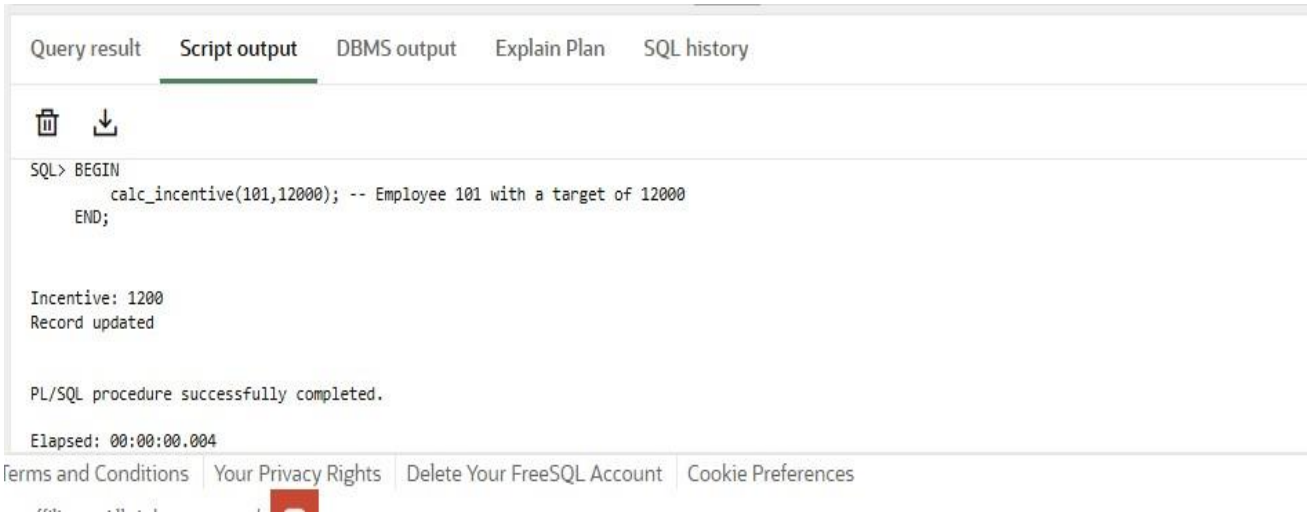


PROGRAM 7

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;
/
```



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
```

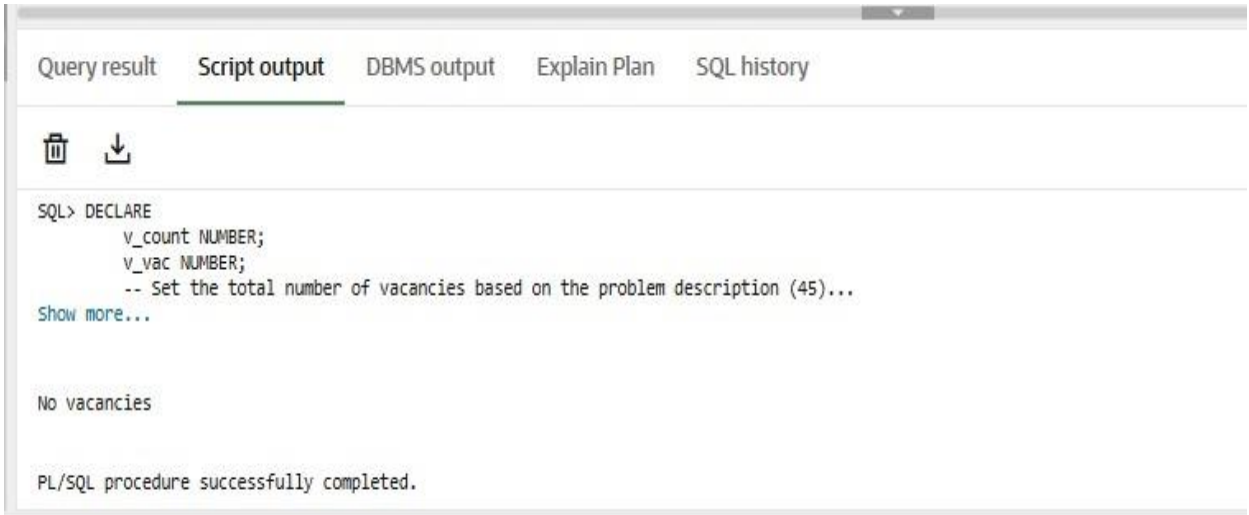
```
   IF p_sales >= 20000 THEN        v_incentive := p_sales *
0.15; -- 15% for sales >= 20,000     ELSIF p_sales >= 10000
THEN        v_incentive := p_sales * 0.10; -- 10% for sales >=
10,000     ELSIF p_sales >= 5000 THEN        v_incentive :=
p_sales * 0.05; -- 5% for sales >= 5,000     ELSE
v_incentive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

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;
```

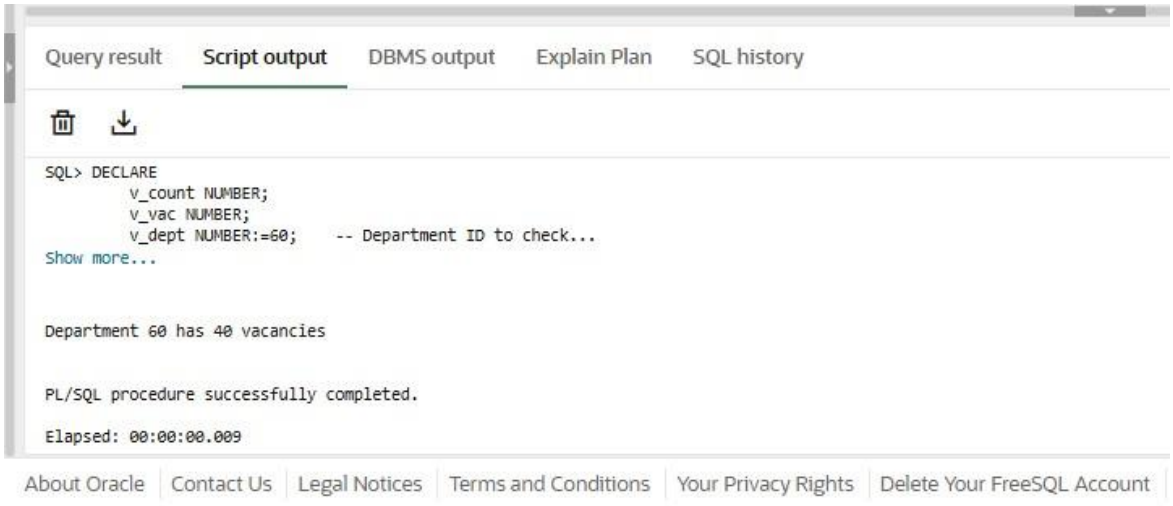
```
DECLARE     v_count
NUMBER;     v_vac
NUMBER;
   -- Set the total number of vacancies based on the problem description (45)
v_total_vacancies CONSTANT NUMBER := 45;
BEGIN
   -- Count employees in department 50
   SELECT COUNT(*) INTO v_count
   FROM HR.employees -- FIX: Schema-qualified the table 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;
/
```



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.

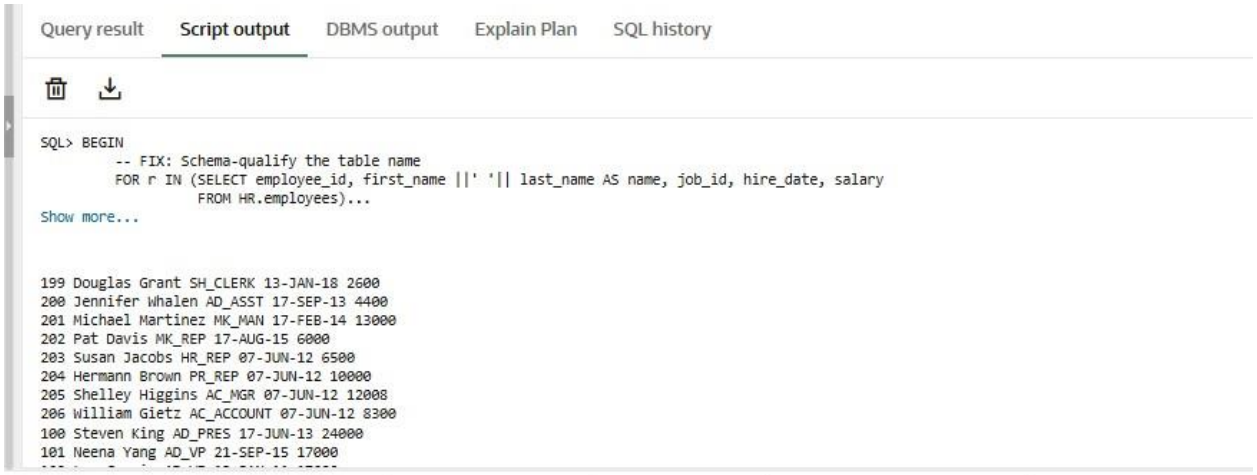
```
SET SERVEROUTPUT ON;

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;
/
```

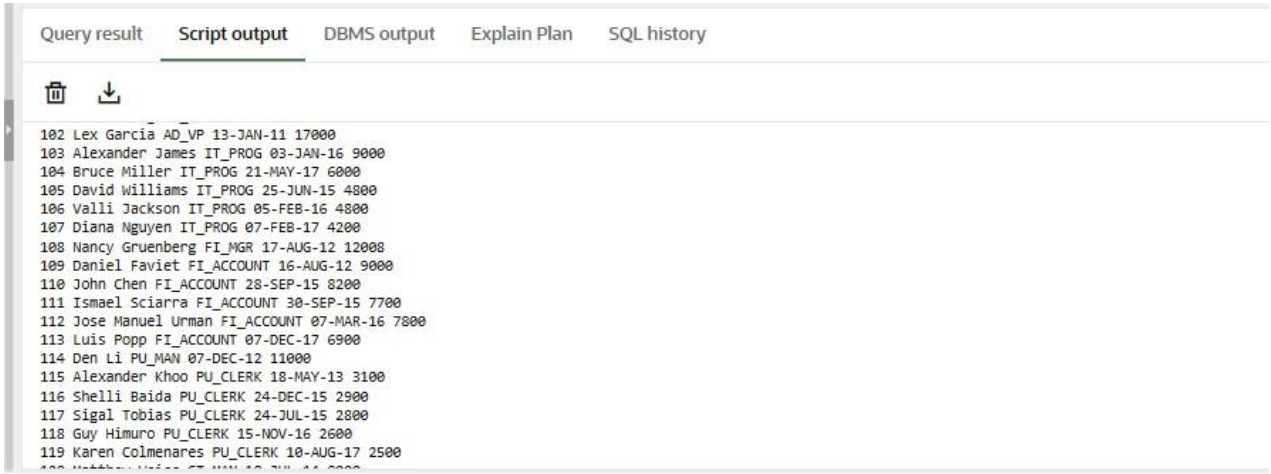


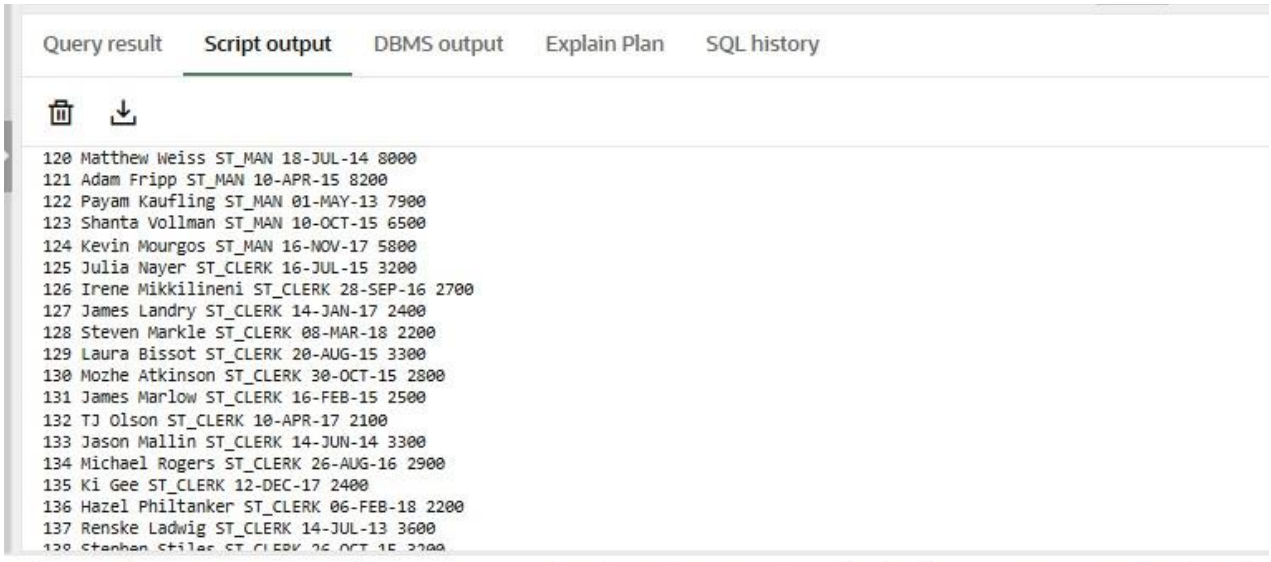
PROGRAM 11

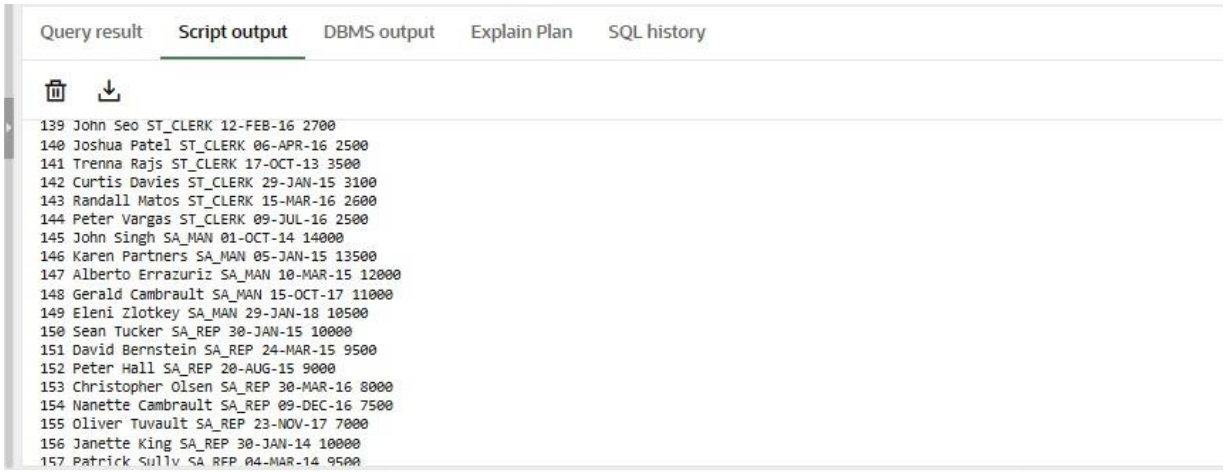
Write a PL/SQL program to display the employee IDs, names, job titles, hire dates, and salaries of all employees.

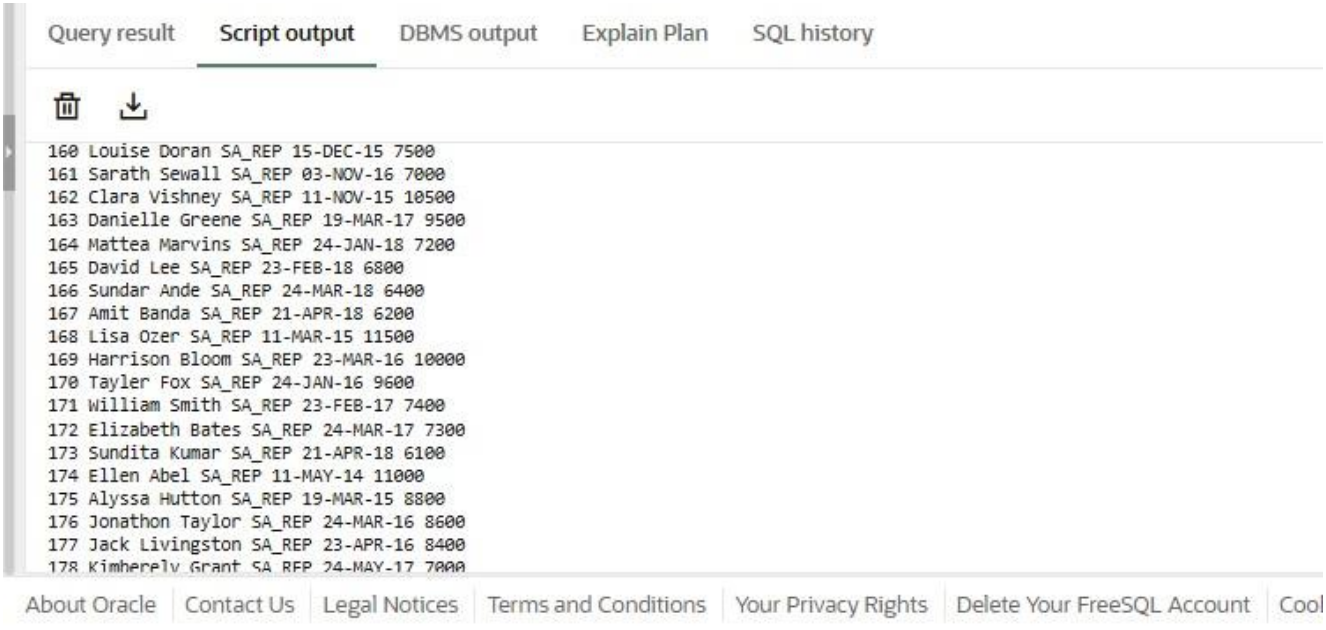
```
SET SERVEROUTPUT ON;


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;
/
```

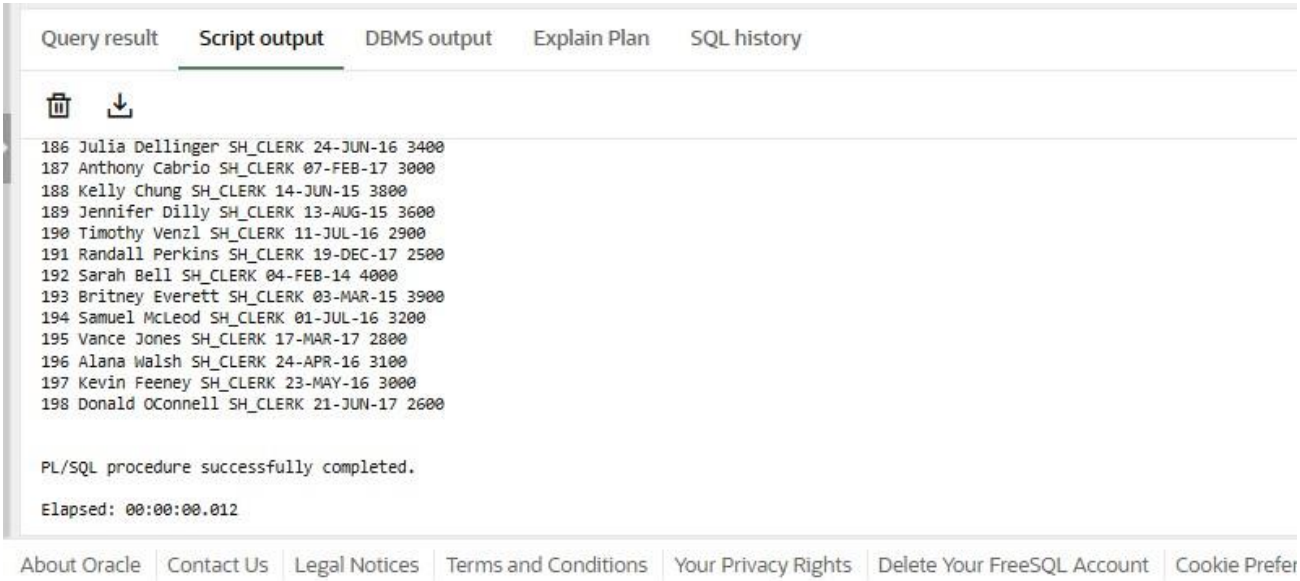







Query result
Script output
DBMS output
Explain Plan
SQL history
139 John Seo ST_CLERK 12-FEB-16 2700
140 Joshua Patel ST_CLERK 06-APR-16 2500
141 Trenna Rajs ST_CLERK 17-OCT-13 3500
142 Curtis Davies ST_CLERK 29-JAN-15 3100
143 Randall Matos ST_CLERK 15-MAR-16 2600
144 Peter Vargas ST_CLERK 09-JUL-16 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-18 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 Tuvault SA_REP 23-NOV-17 7000
156 Janette King SA_REP 30-JAN-14 10000
157 Patrick Sully SA_REP 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 Sarath 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 Kimberely Grant SA_REP 24-MAY-17 7000
About Oracle
Contact Us
Legal Notices
Terms and Conditions
Your Privacy Rights
Delete Your FreeSQL Account
Cool

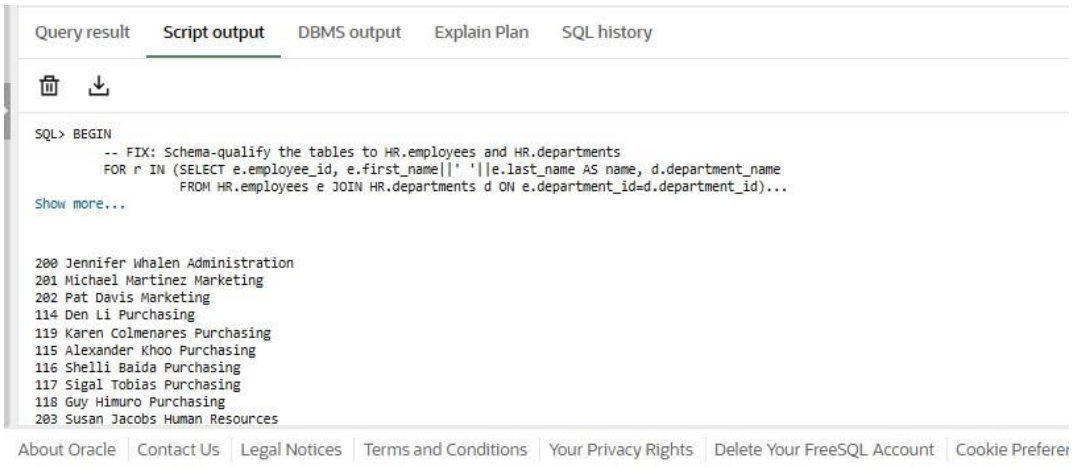

Query result
Script output
DBMS output
Explain Plan
SQL history
186 Julia Dellinger SH_CLERK 24-JUN-16 3400
187 Anthony Cabrio SH_CLERK 07-FEB-17 3000
188 Kelly Chung SH_CLERK 14-JUN-15 3800
189 Jennifer Dilly SH_CLERK 13-AUG-15 3600
190 Timothy Venzl SH_CLERK 11-JUL-16 2900
191 Randall Perkins SH_CLERK 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 OConnell 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 Prefer

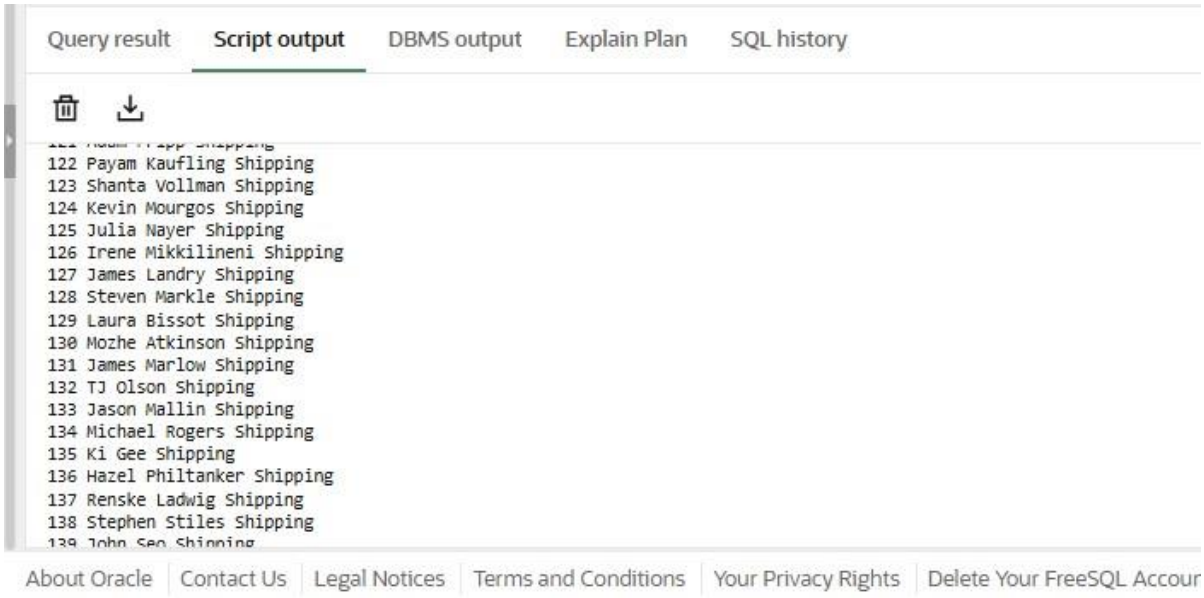
PROGRAM 12

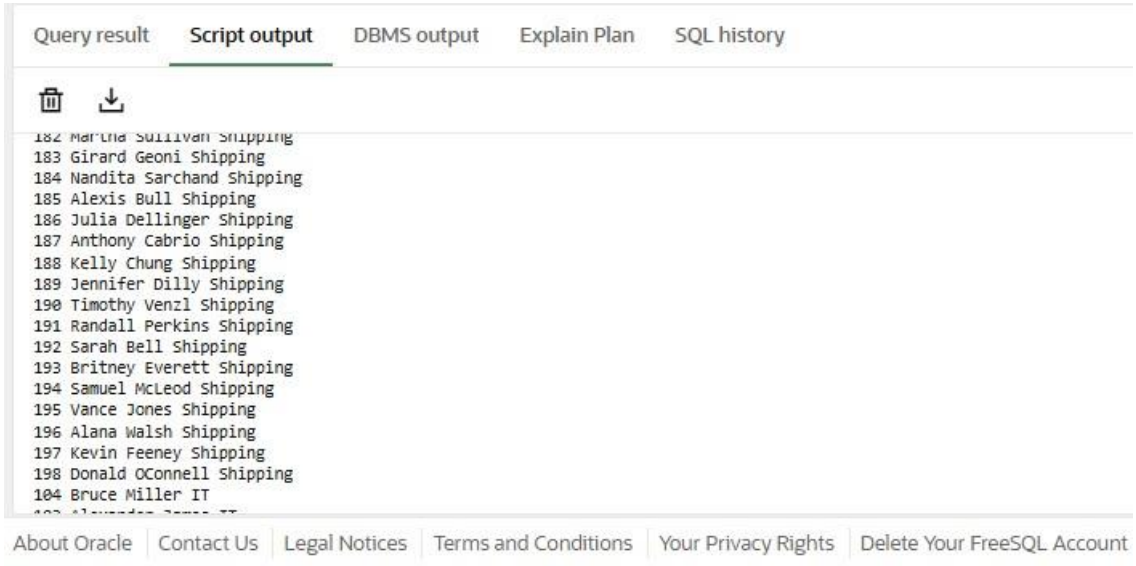
Write a PL/SQL program to display the employee IDs, names, and department names

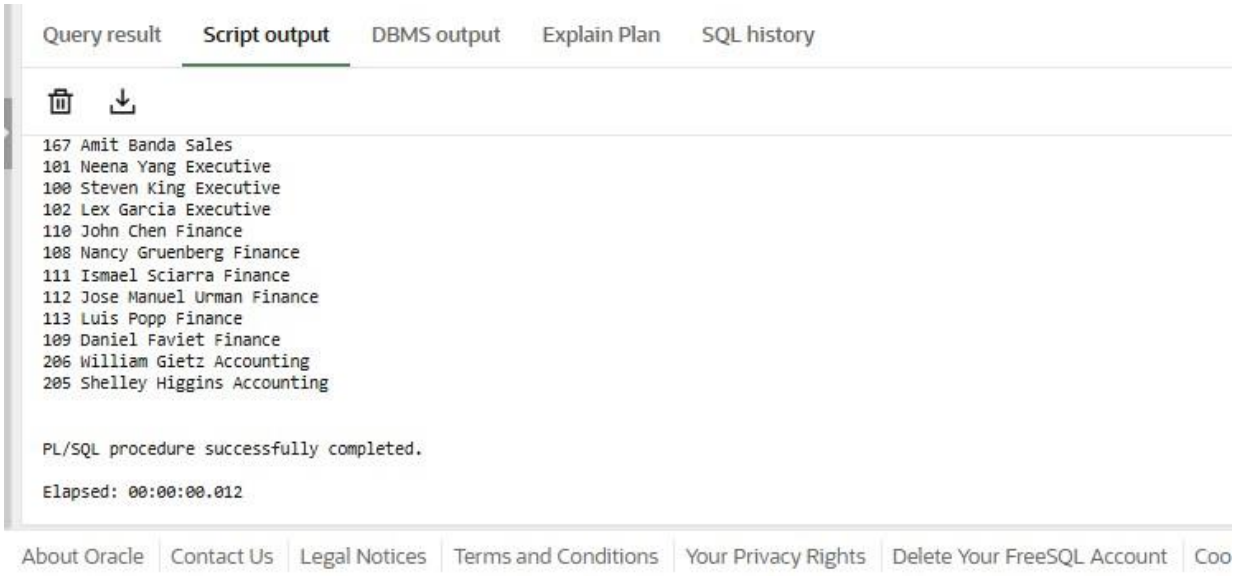
of all employees.

```
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;
/
```




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
About Oracle
Contact Us
Legal Notices
Terms and Conditions
Your Privacy Rights
Delete Your FreeSQL Accour

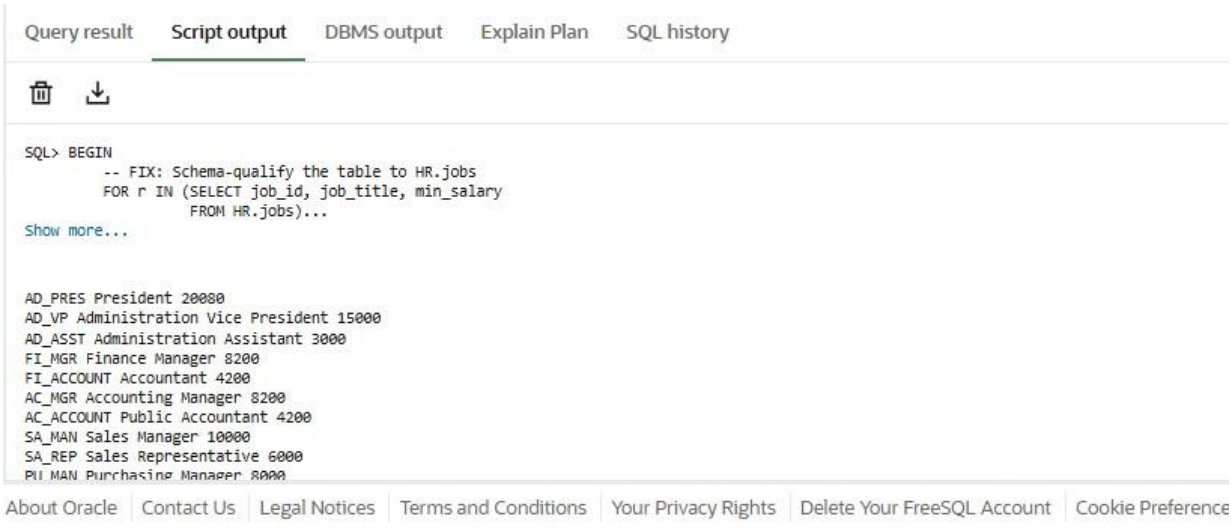

Query result
Script output
DBMS output
Explain Plan
SQL history
183 Girard Geoni Shipping
184 Nandita Sarchand Shipping
185 Alexis Bull Shipping
186 Julia Dellinger Shipping
187 Anthony Cabrio Shipping
188 Kelly Chung Shipping
189 Jennifer Dilly Shipping
190 Timothy Venzl Shipping
191 Randall Perkins 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 OConnell Shipping
104 Bruce Miller IT
About Oracle
Contact Us
Legal Notices
Terms and Conditions
Your Privacy Rights
Delete Your FreeSQL Account

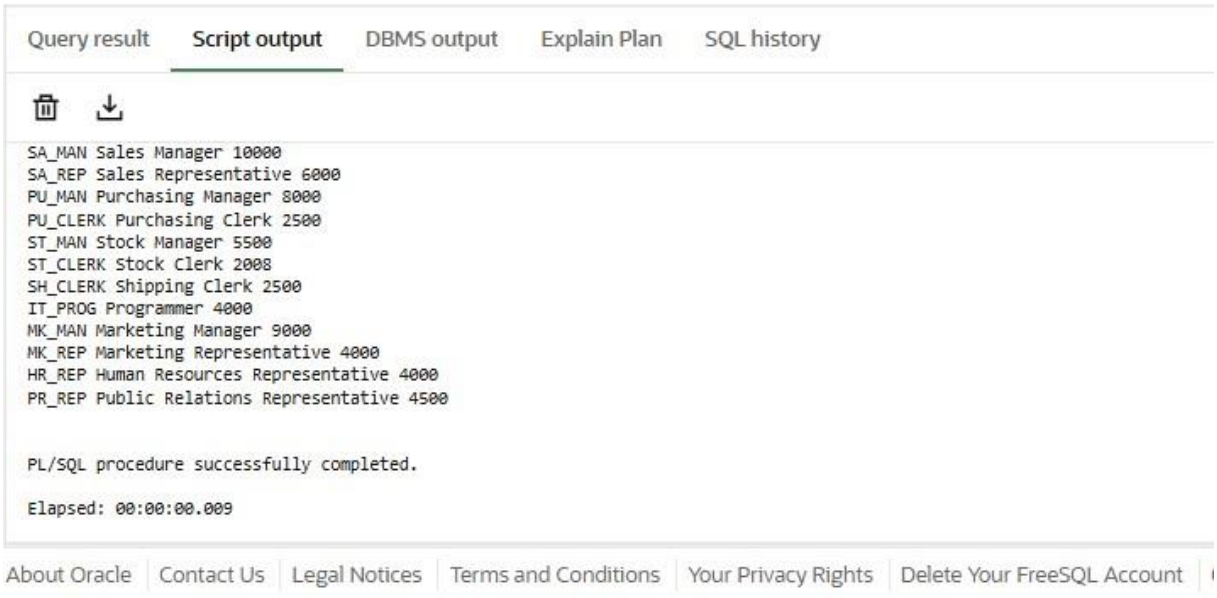


PROGRAM 13

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;
/
```



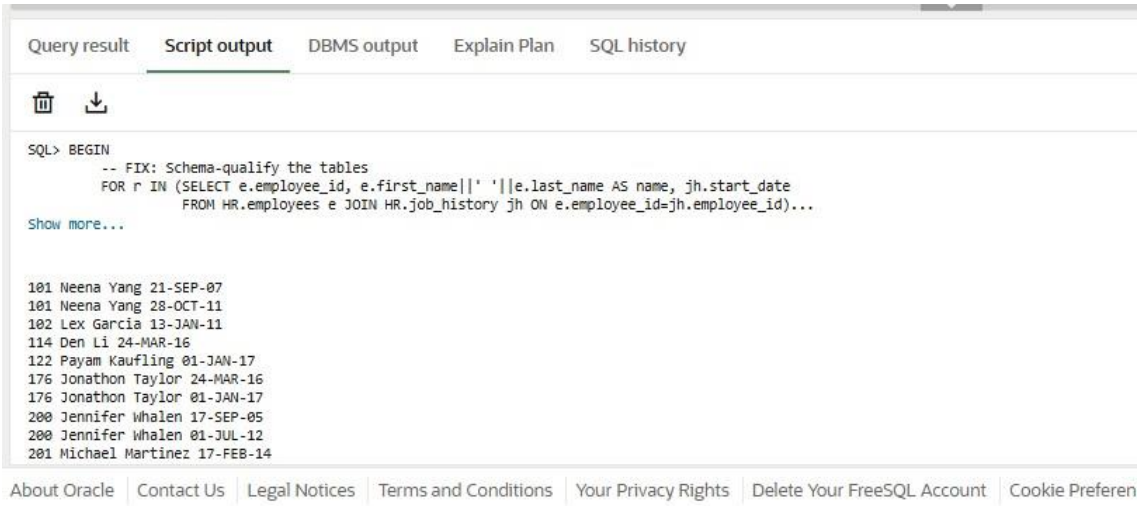


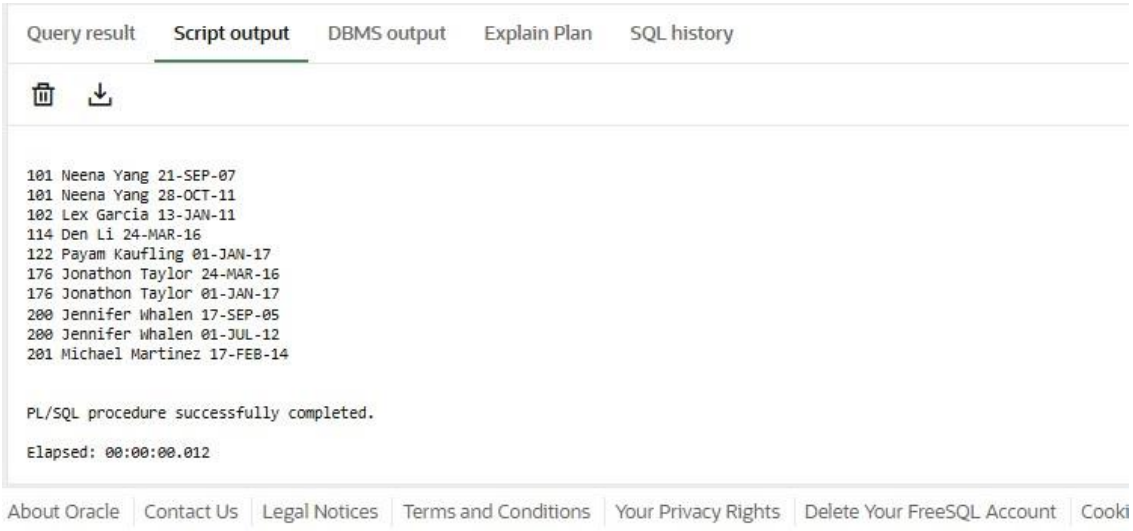
PROGRAM 14

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;
/
```





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
```