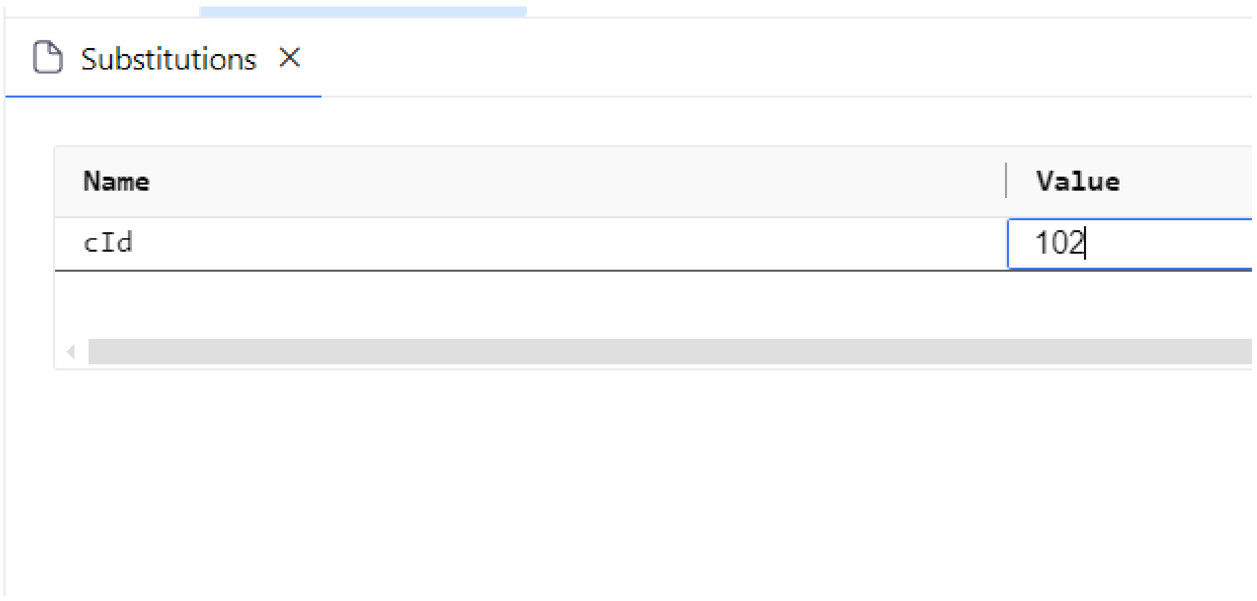
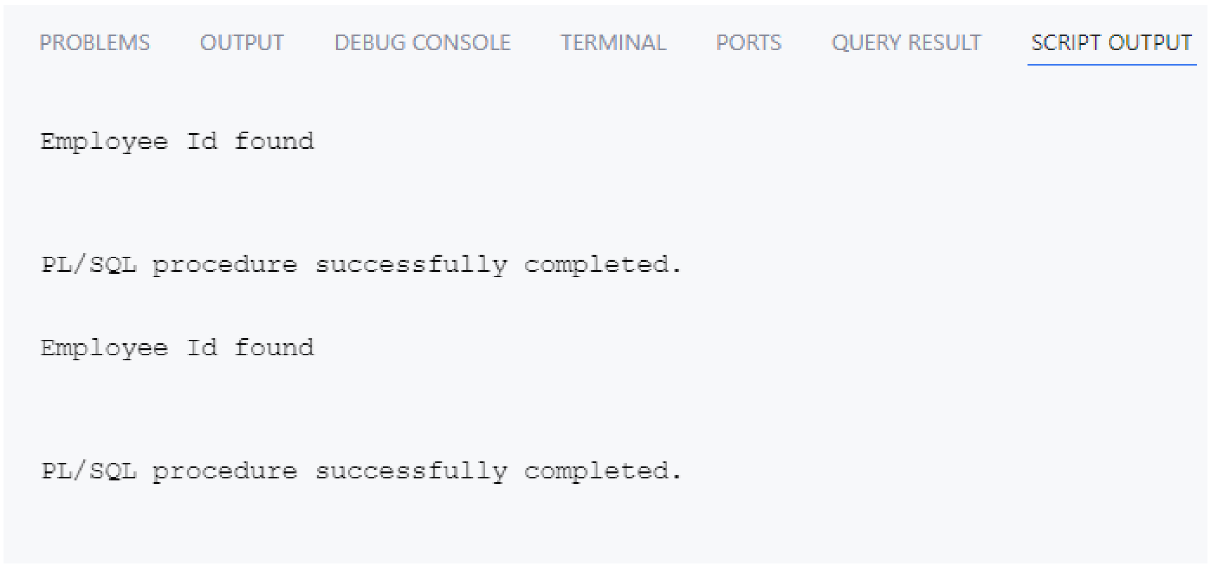
```
--Program to define user defined exception
SELECT * FROM EMPLOYEES;
DECLARE
custom_exception EXCEPTION;
i NUMBER;
empId NUMBER;
BEGIN
    empId := &cId;
    FOR i IN (SELECT * FROM EMPLOYEES) LOOP
        IF i.employee_id = empId THEN
            RAISE custom_exception;
        END IF;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Employee Id not found');
    EXCEPTION
        WHEN custom_exception THEN
            DBMS_OUTPUT.PUT_LINE('Employee Id found');
END;
```

OutPut



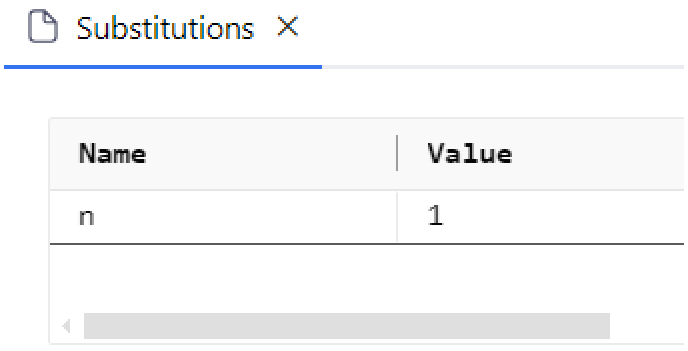
Name	Value
cId	102



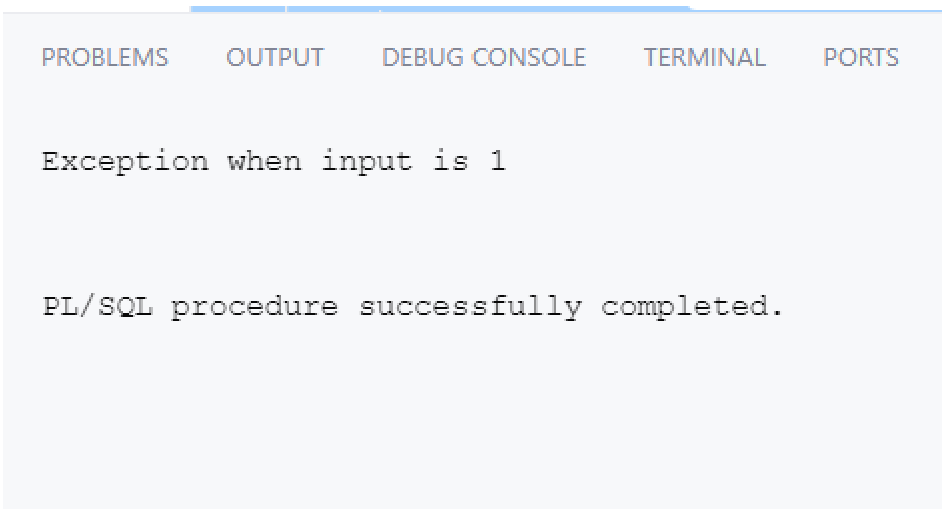
```
Employee Id found


PL/SQL procedure successfully completed.

Employee Id found


PL/SQL procedure successfully completed.
```

```
--Exception Propagation in PL/SQL
DECLARE
    e1 EXCEPTION;
    PRAGMA EXCEPTION_INIT(e1, -20001);
    e2 EXCEPTION;
    PRAGMA EXCEPTION_INIT(e2, -20002);
    e3 EXCEPTION;
    PRAGMA EXCEPTION_INIT(e3, -20003);
    num_input  NUMBER := &n;
BEGIN
    BEGIN -- inner block
        IF num_input = 1 THEN
            RAISE_APPLICATION_ERROR(-20001, 'Exception when 
input is 1');
        ELSIF num_input = 2 THEN
            RAISE_APPLICATION_ERROR(-20002, 'Exception when 
input is 2');
        ELSE
            RAISE_APPLICATION_ERROR(-20003, 'Exception when 
input is not 1 or 2');
        END IF;
    EXCEPTION
        WHEN e1 THEN
            DBMS_OUTPUT.PUT_LINE('Exception when input is 
1');
    END;
    -- Outer Block Exception Handling
    EXCEPTION
        WHEN e2 THEN
            DBMS_OUTPUT.PUT_LINE('Exception when input is 
2');
        WHEN e3 THEN
            DBMS_OUTPUT.PUT_LINE('Exception when input is 
not 1 or 2');
END;
```

Output


Substitutions

Name	Value
n	1

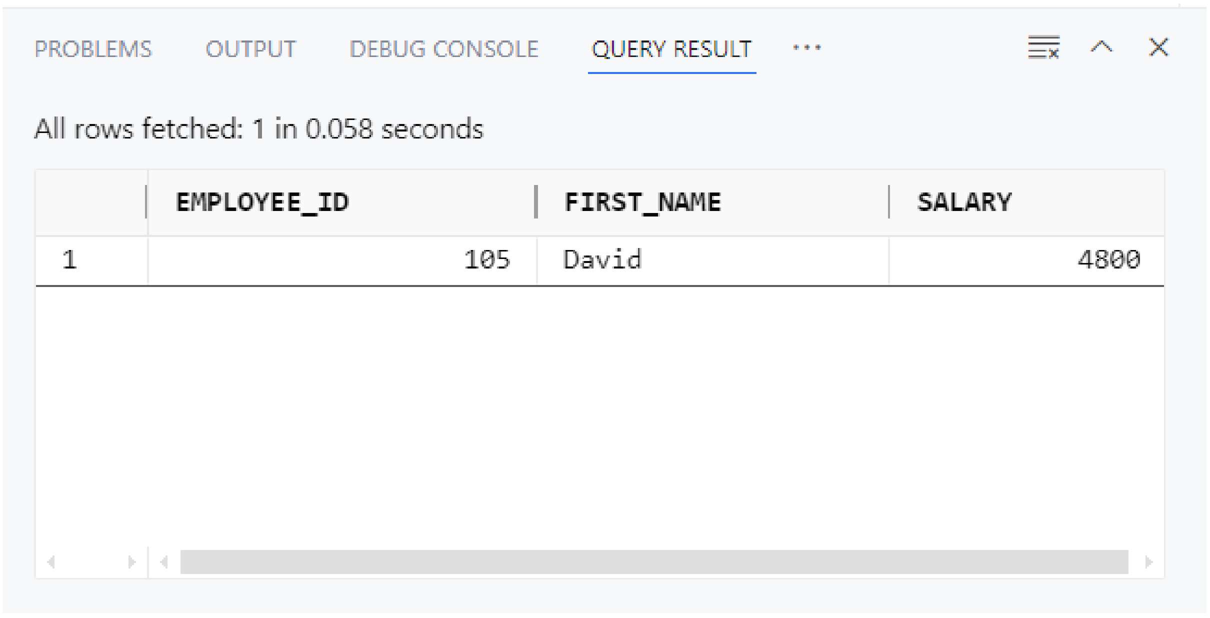


```
Exception when input is 1


PL/SQL procedure successfully completed.
```

```
--Implicit Cursor
DECLARE
    total_rows NUMBER(4);
BEGIN
    -- Perform the update
    UPDATE employees
    SET salary = salary + 500 where employee_id=&emp_id;
    -- Check if the update affected any rows
    IF sql%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('No rows updated.');
    ELSE
        -- If rows were updated, output the number of
affected rows
        total_rows := sql%ROWCOUNT;
        DBMS_OUTPUT.PUT_LINE(total_rows || ' rows
updated.');
    END IF;
END;
/
```

Output

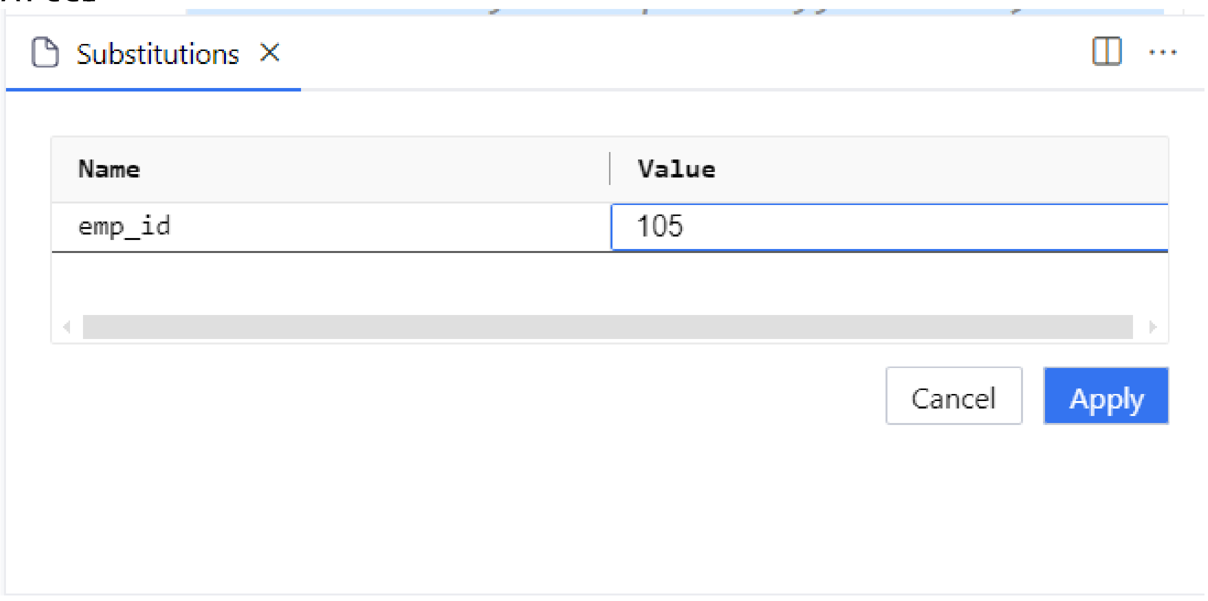
Before



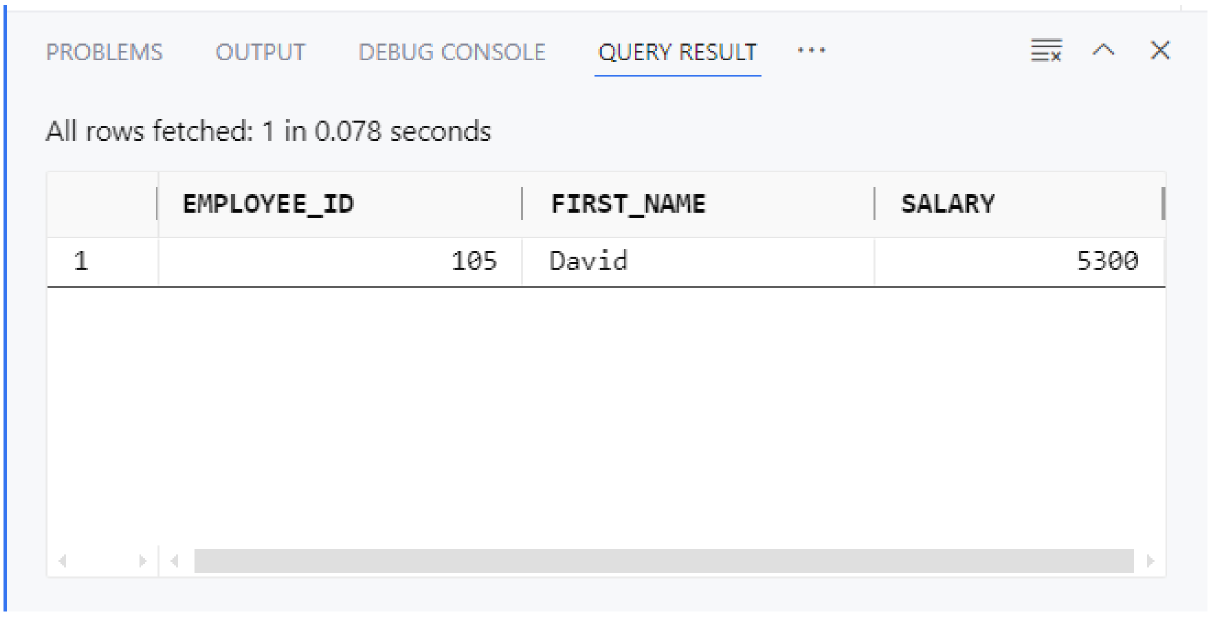
All rows fetched: 1 in 0.058 seconds

	EMPLOYEE_ID	FIRST_NAME	SALARY
1	105	David	4800

After



Name	Value
emp_id	105



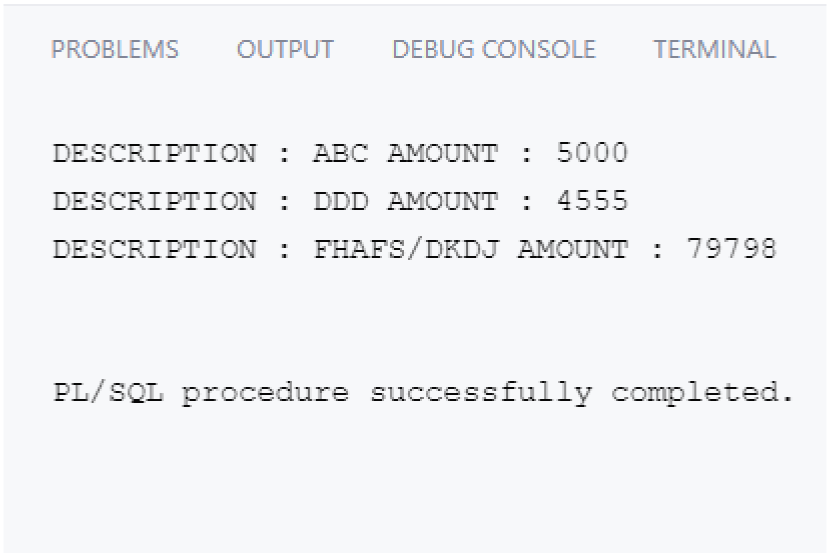
All rows fetched: 1 in 0.078 seconds

	EMPLOYEE_ID	FIRST_NAME	SALARY
1	105	David	5300

```sql
--Create date table
CREATE TABLE DATETABLE (
    ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    DATE1 DATE
);

--Insert data into date table
INSERT INTO DATETABLE (DATE1) VALUES (TO_DATE('2023-02-01',
'YYYY-MM-DD'));

INSERT INTO DATETABLE (DATE1) VALUES (TO_DATE('2023-02-05',
'YYYY-MM-DD'));

INSERT INTO DATETABLE (DATE1) VALUES (TO_DATE('2023-02-10',
'YYYY-MM-DD'));

INSERT INTO DATETABLE (DATE1) VALUES (TO_DATE('2023-02-15',
'YYYY-MM-DD'));

--Create cursor_transaction table
CREATE TABLE CURSOR_TRANSACTION
(TRAN_DATE DATE,
DESCRIPTION VARCHAR2(80),
AMOUNT NUMBER);

--Insert data into cursor_transaction table
INSERT INTO CURSOR_TRANSACTION (TRAN_DATE, DESCRIPTION,
AMOUNT)
VALUES (to_date('01-02-2023', 'dd-mm-yyyy'), 'ABC', 5000);

INSERT INTO CURSOR_TRANSACTION (TRAN_DATE, DESCRIPTION,
AMOUNT)
VALUES (to_date('05-02-2023', 'dd-mm-yyyy'), 'DDD', 4555);
```

```
INSERT INTO CURSOR_TRANSACTION (TRAN_DATE, DESCRIPTION,
AMOUNT)
VALUES (to_date('10-02-2023', 'dd-mm-yyyy'), 'FHAFS/DKDJ',
79798);

INSERT INTO CURSOR_TRANSACTION (TRAN_DATE, DESCRIPTION,
AMOUNT)
VALUES (to_date('15-02-2023', 'dd-mm-yyyy'), 'PAYMENT',
83739);



--cursor loop
DECLARE
  DESCPT  VARCHAR2(80);
  AMT NUMBER;
CURSOR GETDATE IS
    SELECT *
      FROM DATETABLE
     WHERE DATE1 BETWEEN '1-FEB-2023' AND '12-FEB-2023';
  DATE_REC GETDATE%ROWTYPE;

BEGIN

  OPEN GETDATE;
  LOOP
    FETCH GETDATE
      INTO DATE_REC;
    EXIT WHEN GETDATE%NOTFOUND;

   BEGIN
      SELECT DESCRIPTION, AMOUNT
        INTO DESCPT, AMT
        FROM CURSOR_TRANSACTION
       WHERE TRAN_DATE = DATE_REC.DATE1;
      DBMS_OUTPUT.PUT_LINE('DESCRIPTION : ' || DESCPT || '
AMOUNT : ' || AMT);

    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('THERE IS NO DATA FOR DATE : '
|| TO_CHAR(DATE_REC.DATE1,'DD-MON-RRRR'));
   WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('TOO MANY ROWS FOR DATE: ' ||
TO_CHAR(date_rec.DATE1,'DD-MON-RRRR'));

    END;

 END LOOP;
  CLOSE GETDATE;
END;
```

Output

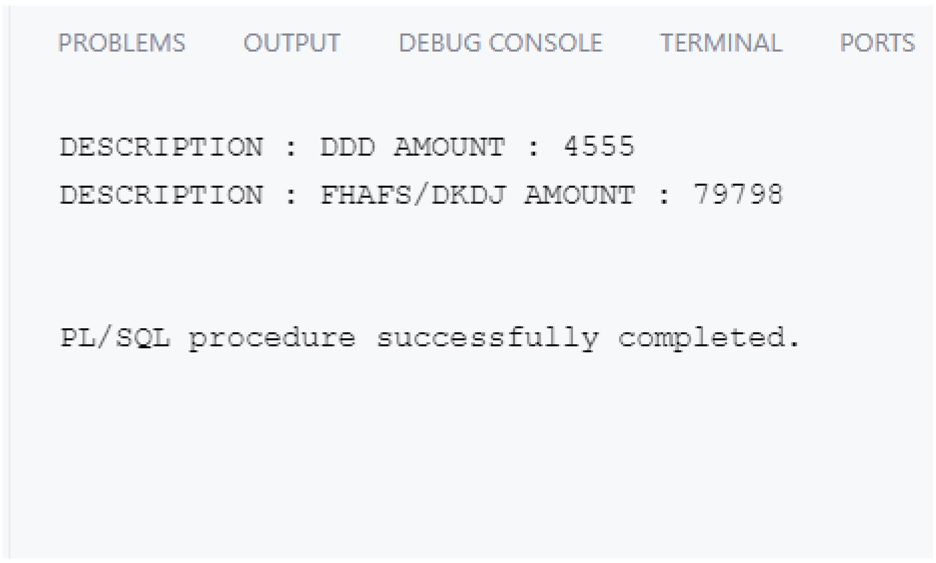


```
DESCRIPTION : ABC AMOUNT : 5000
DESCRIPTION : DDD AMOUNT : 4555
DESCRIPTION : FHAFS/DKDJ AMOUNT : 79798


PL/SQL procedure successfully completed.
```

```
--Nested Cursor
DECLARE
  DE  VARCHAR2(80);
  AMT NUMBER;
  CURSOR GETDATE IS
    SELECT MAX(DATE1)MDATE
      FROM DATETABLE
     WHERE DATE1 BETWEEN '1-FEB-2023' AND '3-FEB-2023';
  MDATE_REC GETDATE%ROWTYPE;
    CURSOR OUTDATE IS
     SELECT *
      FROM DATETABLE
     WHERE DATE1 > MDATE_REC.MDATE AND DATE1<'12-FEB-2023';
   ODATE_REC OUTDATE%ROWTYPE;
BEGIN
  OPEN GETDATE;
  LOOP
    FETCH GETDATE
      INTO MDATE_REC;
    EXIT WHEN GETDATE%NOTFOUND;
      OPEN OUTDATE;
  LOOP
    FETCH OUTDATE
      INTO ODATE_REC;
    EXIT WHEN OUTDATE%NOTFOUND;
   BEGIN
      SELECT DESCRIPTION, AMOUNT
        INTO DE, AMT
        FROM CURSOR_TRANSACTION
       WHERE TRAN_DATE = ODATE_REC.DATE1;
      DBMS_OUTPUT.PUT_LINE('DESCRIPTION : ' || DE || '
AMOUNT : ' || AMT);

    EXCEPTION
      WHEN NO_DATA_FOUND THEN
```

```
        DBMS_OUTPUT.PUT_LINE('THERE ARE NO DATA FOR DATE :
' || TO_CHAR(ODATE_REC.DATE1,'DD-MON-RRRR'));
    END;
  END LOOP;
  CLOSE OUTDATE;
 END LOOP;
  CLOSE GETDATE;
END;
```

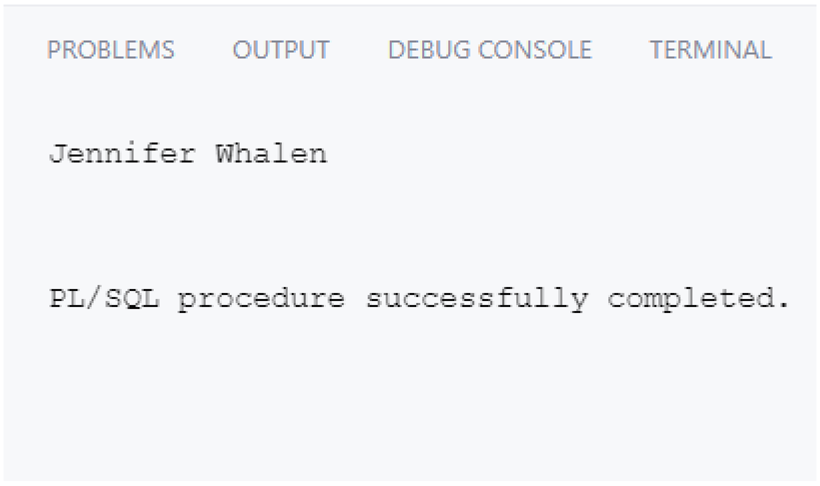
Output



```
DESCRIPTION : DDD AMOUNT : 4555
DESCRIPTION : FHAFS/DKDJ AMOUNT : 79798


PL/SQL procedure successfully completed.
```

```
--Paramaterized Curosor
DECLARE
CURSOR emp_cur (p_dept_id NUMBER) IS
SELECT employee_id, first_name, last_name
FROM employees
WHERE department_id = &p_dept_id;
v_dept_id NUMBER ;
BEGIN
FOR emp_rec IN emp_cur(v_dept_id) LOOP
dbms_output.put_line(emp_rec.first_name || ' ' ||
emp_rec.last_name);
END LOOP;
END;
```

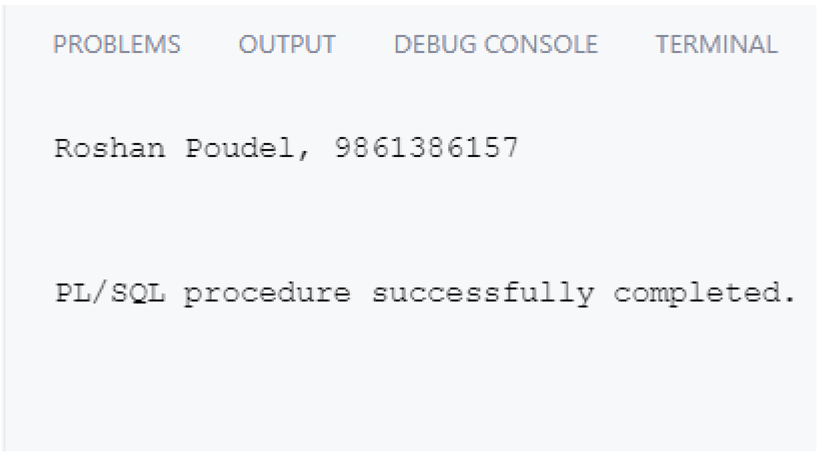
Output



```
Jennifer Whalen


PL/SQL procedure successfully completed.
```

```
--RECORD
DECLARE
TYPE name_rec IS RECORD (
first employees.first_name%TYPE,
last employees.last_name%TYPE
);
TYPE contact IS RECORD (
name name_rec, -- nested record
phone employees.phone_number%TYPE
);
friend contact;
BEGIN
friend.name.first := 'Roshan';
friend.name.last := 'Poudel';
friend.phone := '9861386157';
DBMS_OUTPUT.PUT_LINE (
friend.name.first || ' ' ||
friend.name.last || ', ' ||
friend.phone
);
END;
/
```

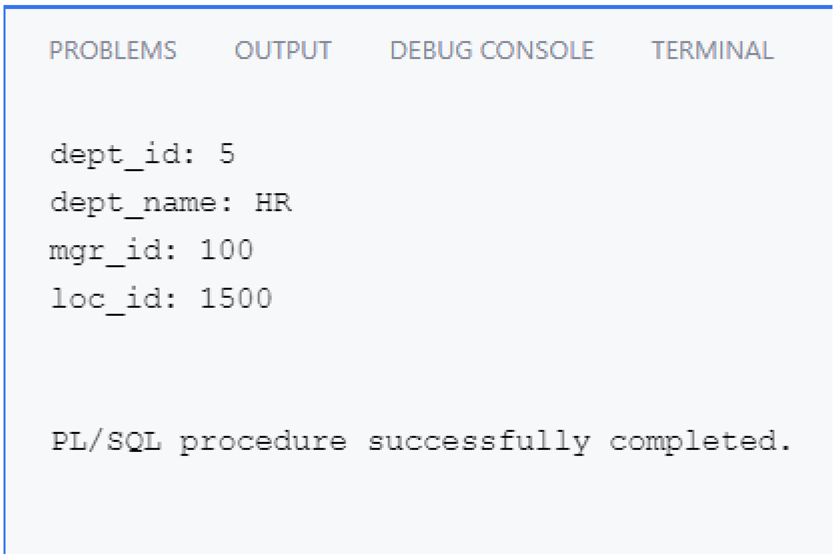
Output



```
Roshan Poudel, 9861386157


PL/SQL procedure successfully completed.
```

```
--user defined record type
DECLARE
TYPE DeptRecTyp IS RECORD (
dept_id NUMBER(4) NOT NULL := 5,
dept_name VARCHAR2(30) NOT NULL := 'HR',
mgr_id NUMBER(6) := 100,
loc_id NUMBER(4) := 1500
);
dept_rec DeptRecTyp;
BEGIN
DBMS_OUTPUT.PUT_LINE('dept_id: ' || dept_rec.dept_id);
DBMS_OUTPUT.PUT_LINE('dept_name: ' || dept_rec.dept_name);
DBMS_OUTPUT.PUT_LINE('mgr_id: ' || dept_rec.mgr_id);
DBMS_OUTPUT.PUT_LINE('loc_id: ' || dept_rec.loc_id);
END;
/
```

Output

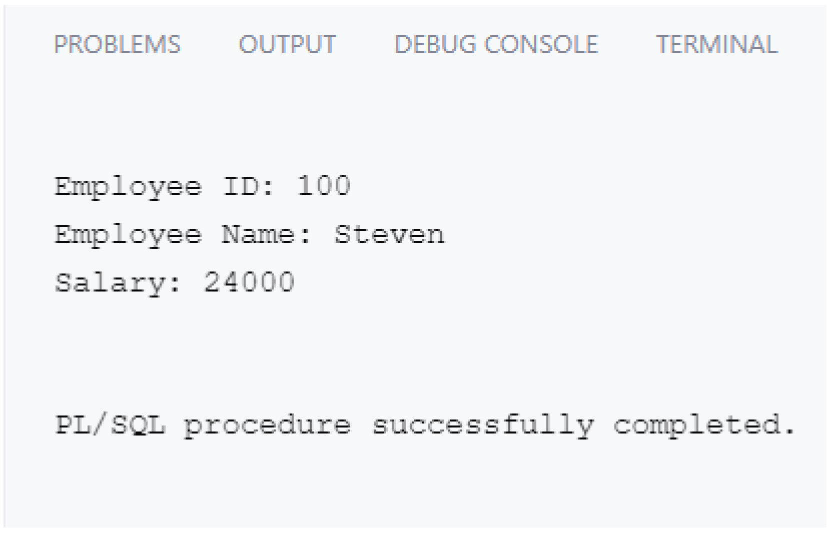


```
dept_id: 5
dept_name: HR
mgr_id: 100
loc_id: 1500


PL/SQL procedure successfully completed.
```

```
DECLARE
  -- Declare variables with %TYPE
  employee_id   employees.employee_id%TYPE;
  employee_name employees.first_name%TYPE;
  salary        employees.salary%TYPE;

  -- Declare the record
  TYPE employee_rec_type IS RECORD (
    employee_id   employees.employee_id%TYPE,
    employee_name employees.first_name%TYPE,
    salary        employees.salary%TYPE
  );

  -- Declare a cursor based on a SELECT statement
  CURSOR emp_cursor IS
    SELECT employee_id, first_name, salary
    FROM employees;
  -- Declare a variable of the record type
  emp_rec employee_rec_type;
BEGIN
  -- Fetch data from the cursor into variables
  OPEN emp_cursor;
  FETCH emp_cursor INTO  employee_id,  employee_name,
salary;
  CLOSE emp_cursor;

  -- Assign values to the fields of the record
  emp_rec.employee_id := employee_id;
  emp_rec.employee_name := employee_name;
  emp_rec.salary := salary;

  -- Display the values of the record fields
  DBMS_OUTPUT.PUT_LINE('Employee ID: ' ||
emp_rec.employee_id);
```

```
  DBMS_OUTPUT.PUT_LINE('Employee Name: ' ||
emp_rec.employee_name);
  DBMS_OUTPUT.PUT_LINE('Salary: ' || emp_rec.salary);
END;
```

Output



```
Employee ID: 100
Employee Name: Steven
Salary: 24000


PL/SQL procedure successfully completed.
```