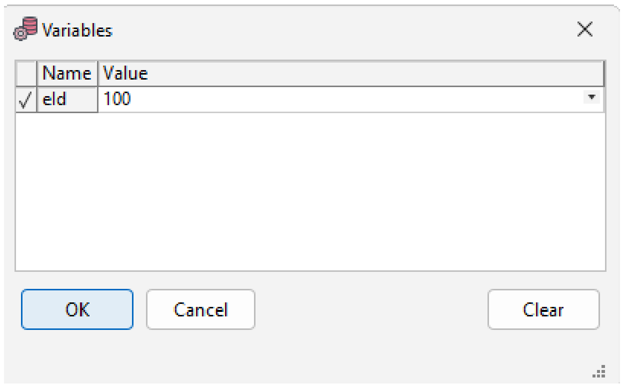
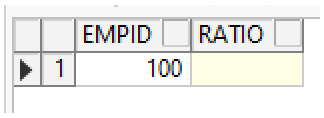


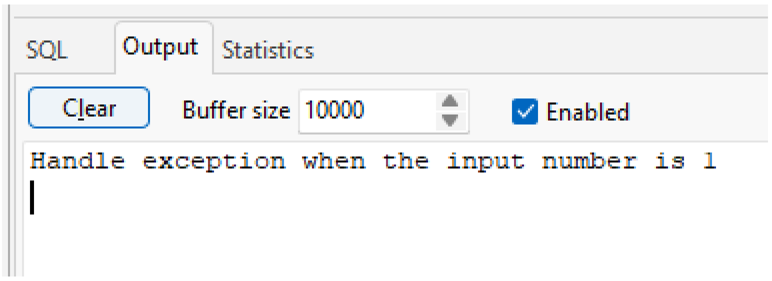
Variables

	Name	Value
✓	eId	100

OK Cancel Clear

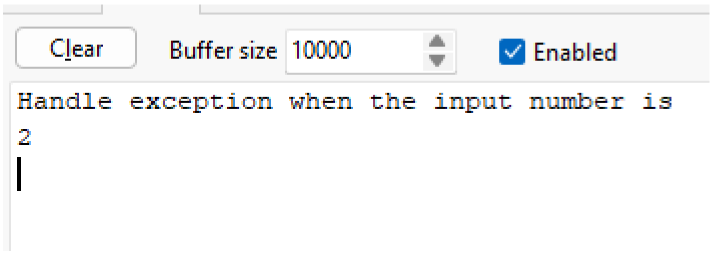


	EMPID	RATIO
1	100	



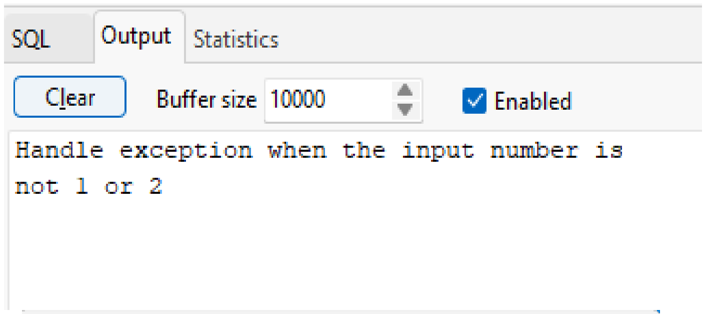
Buffer size 10000 ✓ Enabled

```
Handle exception when the input number is 1
```



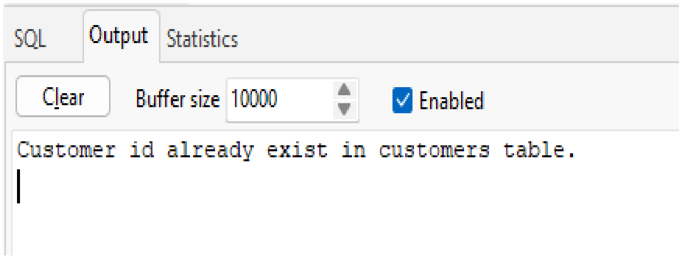
Buffer size 10000 ✓ Enabled

```
Handle exception when the input number is
2
```



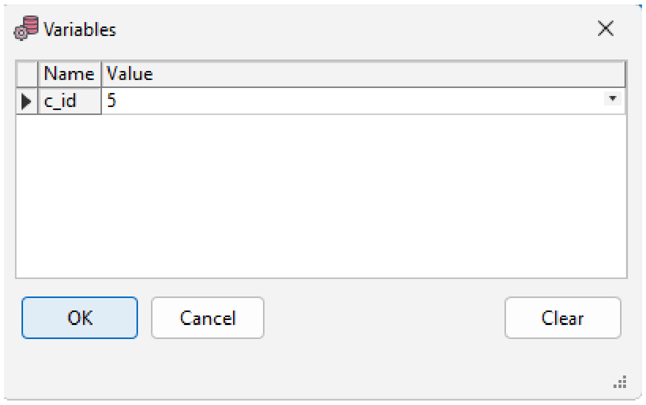
Buffer size 10000 ✓ Enabled

```
Handle exception when the input number is
not 1 or 2
```



Buffer size 10000 ✓ Enabled

```
Customer id already exist in customers table.
```



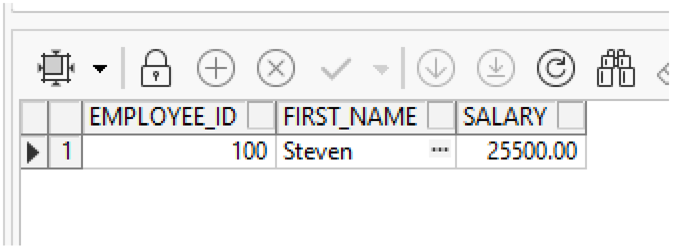
Variables

	Name	Value
▶	c_id	5

OK Cancel Clear

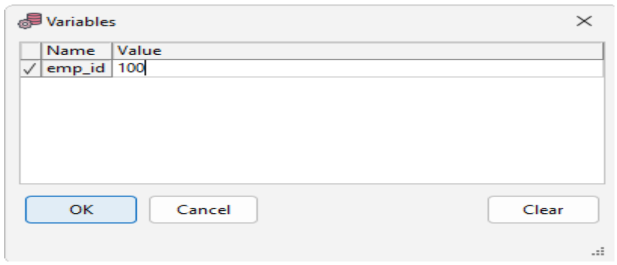
```
DECLARE
   total_rows NUMBER(4);
BEGIN
   -- Perform the update
   UPDATE employees
   SET salary = salary + 500 where employee_id=&emp_id;
   -- Check if the update affected any rows
   IF sql%NOTFOUND THEN
      DBMS_OUTPUT.PUT_LINE('No rows updated.');
   ELSE
      -- If rows were updated, output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

Before



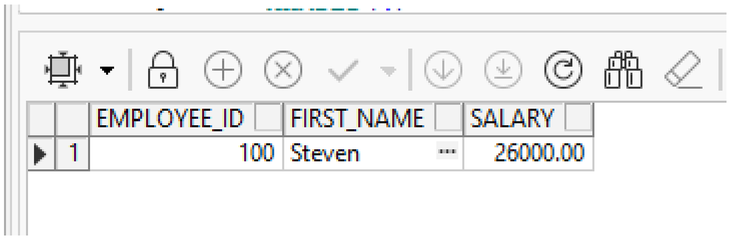
	EMPLOYEE_ID	FIRST_NAME	SALARY
1	100	Steven	25500.00

After

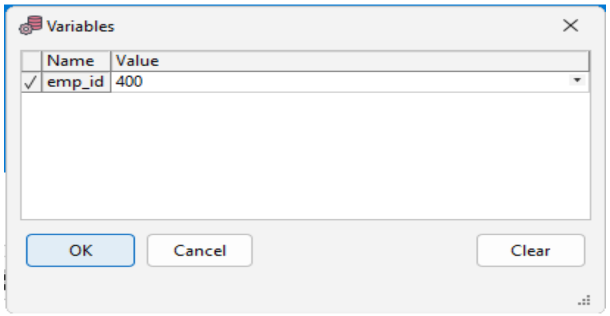


Variables

Name	Value
✓ emp_id	100

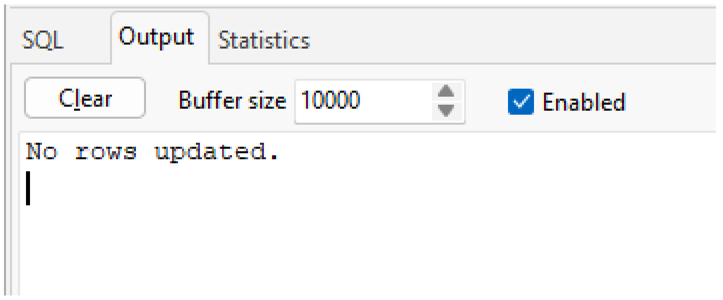


	EMPLOYEE_ID	FIRST_NAME	SALARY
1	100	Steven	26000.00



Variables

Name	Value
✓ emp_id	400



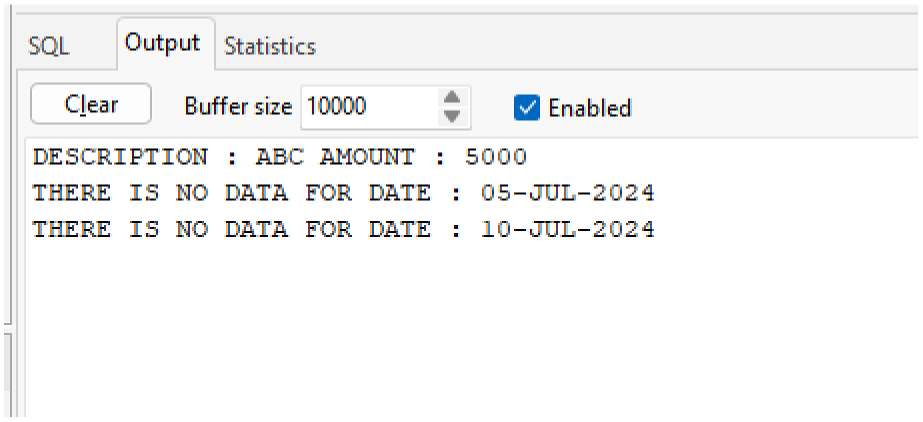
Buffer size 10000 ✓ Enabled

```
No rows updated.
```

```
CREATE TABLE DATETABLE (
    ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    DATE1 DATE
);
INSERT INTO DATETABLE (DATE1) VALUES (TO_DATE('2024-07-01', 'YYYY-MM-DD'));
INSERT INTO DATETABLE (DATE1) VALUES (TO_DATE('2024-07-05', 'YYYY-MM-DD'));
INSERT INTO DATETABLE (DATE1) VALUES (TO_DATE('2024-07-10', 'YYYY-MM-DD'));
INSERT INTO DATETABLE (DATE1) VALUES (TO_DATE('2024-07-15', 'YYYY-MM-DD'));

CREATE TABLE CURSOR_TRANSACTION
(TRAN_DATE DATE,
DESCRIPTION VARCHAR2(80),
AMOUNT NUMBER);
insert into CURSOR_TRANSACTION (TRAN_DATE, DESCRIPTION, AMOUNT)
values (to_date('01-07-2024', 'dd-mm-yyyy'), 'ABC', 5000);

insert into CURSOR_TRANSACTION (TRAN_DATE, DESCRIPTION, AMOUNT)
values (to_date('03-07-2024', 'dd-mm-yyyy'), 'DDD', 4555);

insert into CURSOR_TRANSACTION (TRAN_DATE, DESCRIPTION, AMOUNT)
values (to_date('04-07-2024', 'dd-mm-yyyy'), 'FHAFS/DKDJ', 79798);

insert into CURSOR_TRANSACTION (TRAN_DATE, DESCRIPTION, AMOUNT)
values (to_date('09-07-2024', 'dd-mm-yyyy'), 'PAYMENT', 83739);

COMMIT;

DECLARE
  DESCPT  VARCHAR2(80);
  AMT NUMBER;
```

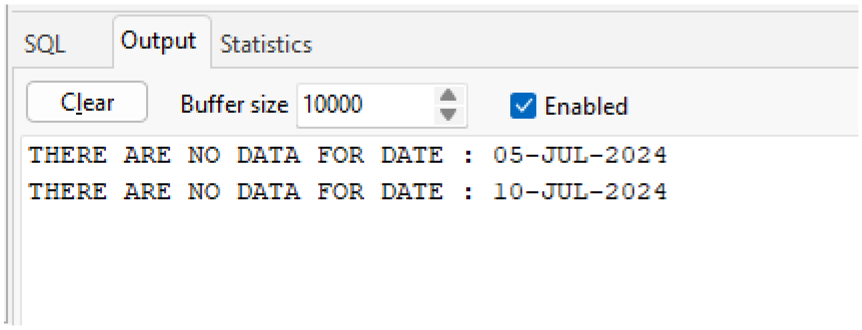
```
- Cursor with loop
CURSOR GETDATE IS
    SELECT *
      FROM DATETABLE
     WHERE DATE1 BETWEEN '1-JUL-2024' AND '11-JUL-2024';
  DATE_REC GETDATE%ROWTYPE;

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;
```



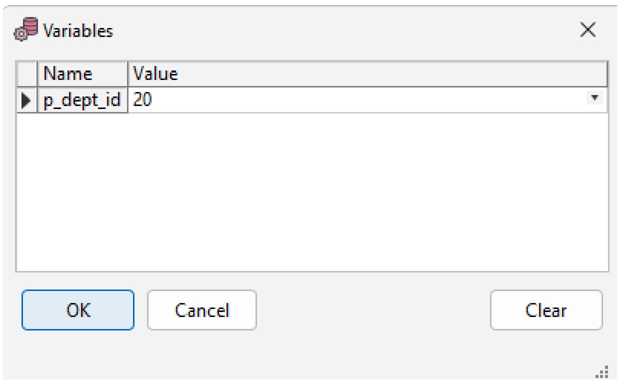
```
DESCRIPTION : ABC AMOUNT : 5000
THERE IS NO DATA FOR DATE : 05-JUL-2024
THERE IS NO DATA FOR DATE : 10-JUL-2024
```

```
–Nested Curosr
DECLARE
  DE  VARCHAR2(80);
  AMT NUMBER;



  CURSOR GETDATE IS
    SELECT MAX(DATE1)MDATE
      FROM DATETABLE
     WHERE DATE1 BETWEEN '1-JUL-2024' AND '3-JUL-2024';
  MDATE_REC GETDATE%ROWTYPE;

    CURSOR OUTDATE IS
     SELECT *
      FROM DATETABLE
     WHERE DATE1 > MDATE_REC.MDATE AND DATE1<'12-JUL-2024';
   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;
```

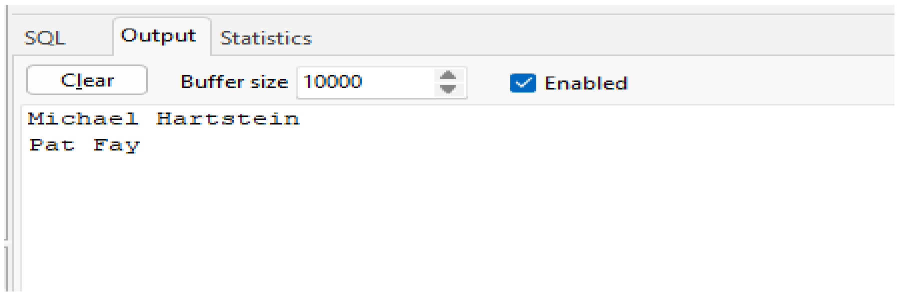


```
THERE ARE NO DATA FOR DATE : 05-JUL-2024
THERE ARE NO DATA FOR DATE : 10-JUL-2024
```

```
–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;
```

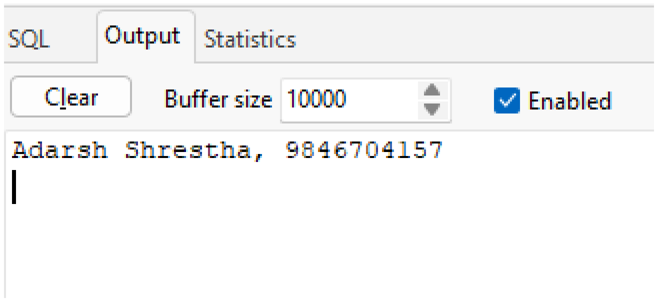


Name	Value
p_dept_id	20



```
Michael Hartstein
Pat Fay
```

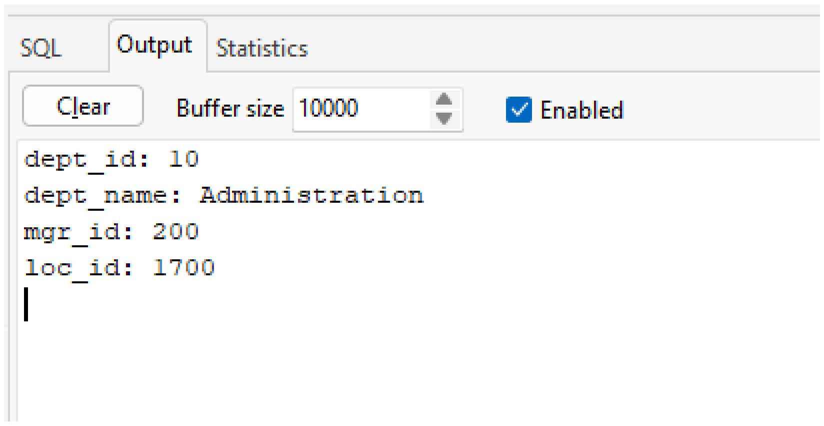
```
DECLARE
TYPE name_rec IS RECORD (
first employees.first_name%TYPE,
last employees.last_name%TYPE
);
TYPE contact IS RECORD (
name name_rec, -- nested record
phone employees.phone_number%TYPE
);
friend contact;
BEGIN
friend.name.first := 'Adarsh';
friend.name.last := 'Shrestha';
friend.phone := '9846704157';
DBMS_OUTPUT.PUT_LINE (
friend.name.first || ' ' ||
friend.name.last || ', ' ||
friend.phone
);
END;
/
```



Buffer size 10000 ✓ Enabled

```
Adarsh Shrestha, 9846704157
```

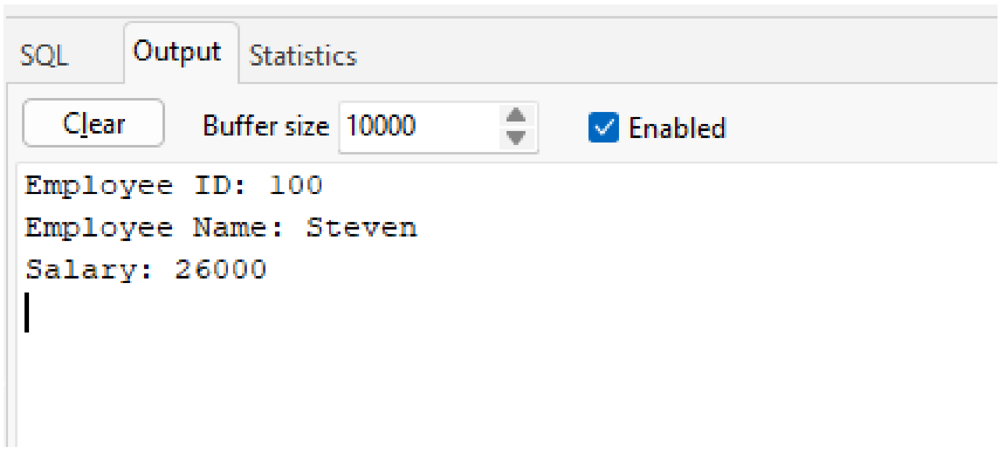
```
DECLARE
TYPE DeptRecTyp IS RECORD (
dept_id NUMBER(4) NOT NULL := 10,
dept_name VARCHAR2(30) NOT NULL := 'Administration',
mgr_id NUMBER(6) := 200,
loc_id NUMBER(4) := 1700
);
dept_rec DeptRecTyp;
BEGIN
DBMS_OUTPUT.PUT_LINE('dept_id: ' || dept_rec.dept_id);
DBMS_OUTPUT.PUT_LINE('dept_name: ' || dept_rec.dept_name);
DBMS_OUTPUT.PUT_LINE('mgr_id: ' || dept_rec.mgr_id);
DBMS_OUTPUT.PUT_LINE('loc_id: ' || dept_rec.loc_id);
END;
/
```



Buffer size 10000 ✓ Enabled

```
dept_id: 10
dept_name: Administration
mgr_id: 200
loc_id: 1700
```

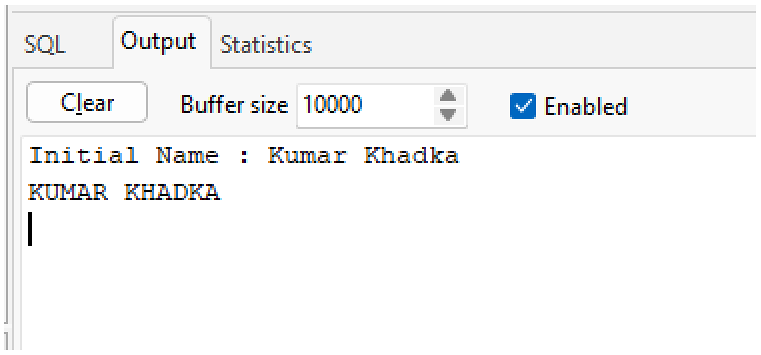
```
DECLARE
  -- Declare variables with %TYPE
  employee_id   employees.employee_id%TYPE;
  employee_name employees.first_name%TYPE;
  salary        employees.salary%TYPE;

  -- 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;
```



```
Employee ID: 100
Employee Name: Steven
Salary: 26000
```

Anonymous Procedure

```
DECLARE -- declare variables and subprograms
  fname    VARCHAR2(20) := 'Kumar';
  lname    VARCHAR2(25) := 'Khadka';

-- declare a local procedure which can only be used in this block
  PROCEDURE upper_name ( v1 IN OUT VARCHAR2, v2 IN OUT VARCHAR2) AS
    BEGIN
      v1 := UPPER(v1); -- change the string to uppercase
      v2 := UPPER(v2); -- change the string to uppercase
    END upper_name;

-- start of executable part of block
BEGIN
  DBMS_OUTPUT.PUT_LINE('Initial Name :'||' '||fname || ' ' || lname ); -- display initial values
  upper_name (fname, lname); -- call the procedure with parameters
  DBMS_OUTPUT.PUT_LINE(fname || ' ' || lname ); -- display new values
END;
```

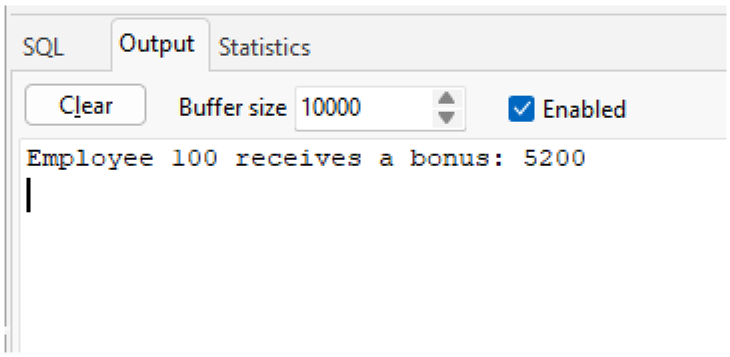


Buffer size 10000   ✓ Enabled

```
Initial Name : Kumar Khadka
KUMAR KHADKA
```

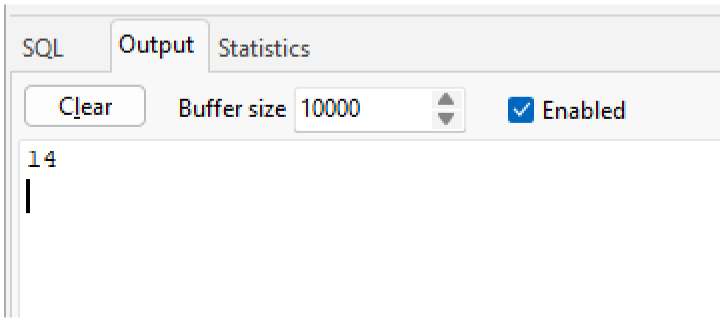
```
CREATE OR REPLACE PROCEDURE award_bonus (emp_id IN NUMBER, bonus_rate IN
NUMBER)
 AS
-- declare variables to hold values from table columns, use %TYPE attribute
  emp_comm       employees.commission_pct%TYPE;
  emp_sal        employees.salary%TYPE;
-- declare an exception to catch when the salary is NULL
  salary_missing  EXCEPTION;
BEGIN  -- executable part starts here
-- select the column values into the local variables
  SELECT salary, commission_pct INTO emp_sal, emp_comm
  FROM employees
   WHERE employee_id = emp_id;
-- check whether the salary for the employee is null, if so, raise an exception
  IF emp_sal IS NULL THEN
    RAISE salary_missing;
ELSE
    IF emp_comm IS NULL THEN
-- if this is not a commissioned employee, increase the salary by the bonus rate
-- for this example, do not make the actual update to the salary
-- UPDATE employee SET salary = salary + salary * bonus_rate
--   WHERE employee_id = emp_id;
     DBMS_OUTPUT.PUT_LINE('Employee ' || emp_id || ' receives a bonus: '
                   || TO_CHAR(emp_sal * bonus_rate) );
    ELSE
     DBMS_OUTPUT.PUT_LINE('Employee ' || emp_id
                   || ' receives a commission. No bonus allowed.');
    END IF;
  END IF;
EXCEPTION  -- exception-handling part starts here
WHEN salary_missing THEN
```

```
      DBMS_OUTPUT.PUT_LINE('Employee ' || emp_id ||
                           ' does not have a value for salary. No update.');
   WHEN OTHERS THEN
      NULL; -- for other exceptions do nothing
END award_bonus;
/



BEGIN
  award_bonus(100,0.2);
END;
```



```
Employee 100 receives a bonus: 5200
```

```
—Out Procedure
CREATE OR REPLACE PROCEDURE calculate_sum(
num1 IN NUMBER,
num2 IN NUMBER,
sum OUT NUMBER
)
IS
BEGIN
sum := num1 + num2;
END;

DECLARE
num1 NUMBER;
num2 NUMBER;
sum_num NUMBER;
BEGIN
  calculate_sum(5,9,sum_num);
  DBMS_OUTPUT.PUT_LINE(sum_num);
END;
/
```



```
14
```