**SECTION 1**

CREATE OR REPLACE PROCEDURE print\_date IS

v\_date VARCHAR2(30);

BEGIN

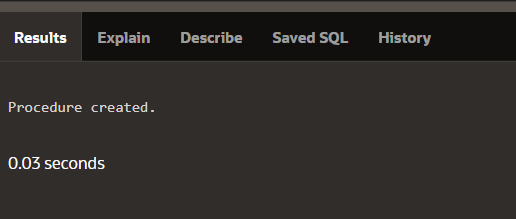
SELECT TO\_CHAR(SYSDATE,'Mon DD, YYYY')

INTO v\_date

FROM DUAL;

DBMS\_OUTPUT.PUT\_LINE(v\_date);

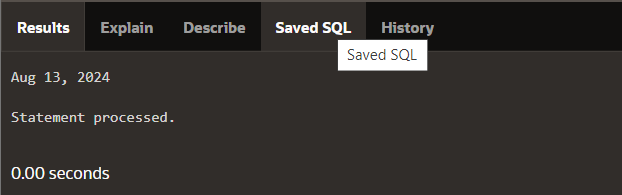
END;



BEGIN

PRINT\_DATE;

END;



CREATE OR REPLACE FUNCTION tomorrow (p\_today IN DATE)

RETURN DATE IS

v\_tomorrow DATE;

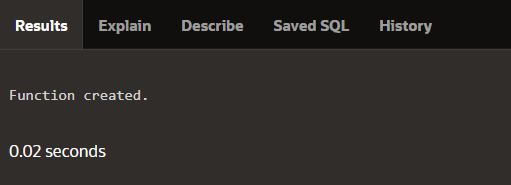
BEGIN

SELECT p\_today + 1 INTO v\_tomorrow

FROM DUAL;

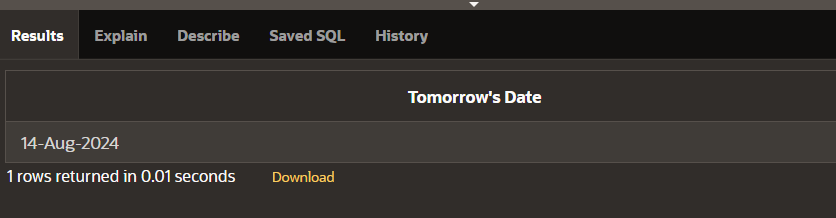
RETURN v\_tomorrow;

END;



SELECT TOMORROW(SYSDATE) AS "Tomorrow's Date"

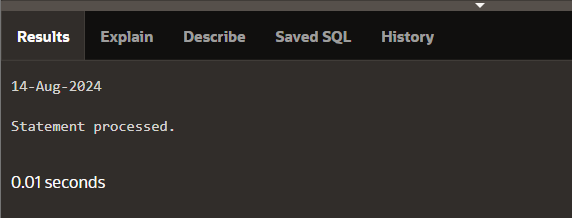
FROM DUAL;



BEGIN

DBMS\_OUTPUT.PUT\_LINE(TOMORROW(SYSDATE));

END;



DECLARE

v\_emp\_count NUMBER;

BEGIN

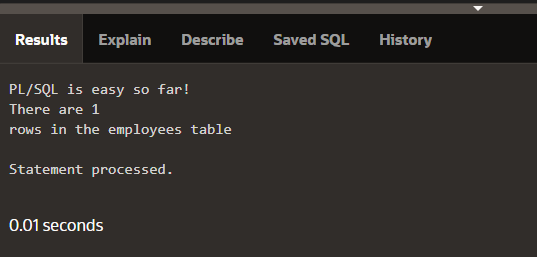
DBMS\_OUTPUT.PUT\_LINE('PL/SQL is easy so far!');

SELECT COUNT(\*) INTO v\_emp\_count FROM dual;

DBMS\_OUTPUT.PUT\_LINE('There are '||v\_emp\_count||'

rows in the employees table');

END;



**SECTION 2**

DECLARE

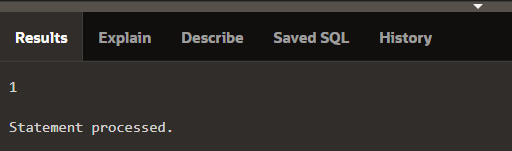
v\_counter INTEGER := 0;

BEGIN

v\_counter := v\_counter + 1;

DBMS\_OUTPUT.PUT\_LINE(v\_counter);

END;



DECLARE

v\_myname VARCHAR2(20);

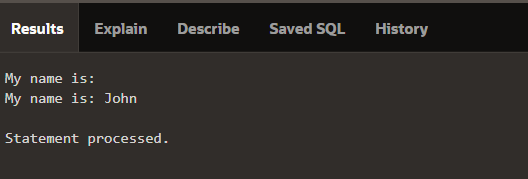
BEGIN

DBMS\_OUTPUT.PUT\_LINE('My name is: '||v\_myname);

v\_myname := 'John';

DBMS\_OUTPUT.PUT\_LINE('My name is: '||v\_myname);

END;



DECLARE

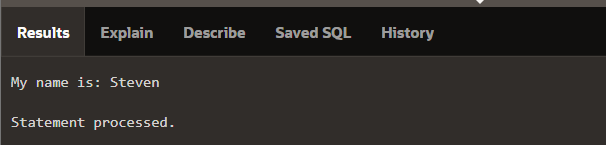
v\_myname VARCHAR2(20):= 'John';

BEGIN

v\_myname := 'Steven';

DBMS\_OUTPUT.PUT\_LINE('My name is: '|| v\_myname);

END;



DECLARE

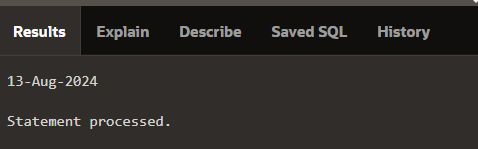
v\_date VARCHAR2(30);

BEGIN

SELECT TO\_CHAR(SYSDATE) INTO v\_date FROM DUAL;

DBMS\_OUTPUT.PUT\_LINE(v\_date);

END;



CREATE FUNCTION num\_characters (p\_string IN VARCHAR2)

RETURN INTEGER IS

v\_num\_characters INTEGER;

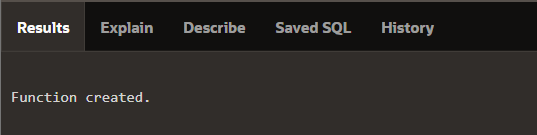
BEGIN

SELECT LENGTH(p\_string) INTO v\_num\_characters

FROM DUAL;

RETURN v\_num\_characters;

END;



DECLARE

v\_length\_of\_string INTEGER;

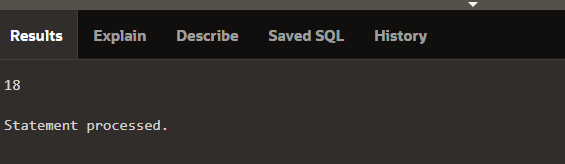
BEGIN

v\_length\_of\_string := num\_characters('Oracle

Corporation');

DBMS\_OUTPUT.PUT\_LINE(v\_length\_of\_string);

END;



CREATE PROCEDURE datee IS

v\_date VARCHAR2(30);

BEGIN

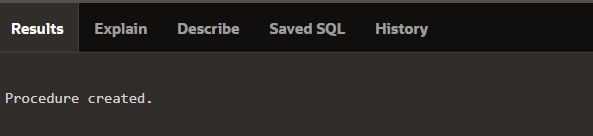
SELECT TO\_CHAR(SYSDATE,'Mon DD, YYYY')

INTO v\_date

FROM DUAL;

DBMS\_OUTPUT.PUT\_LINE(v\_date);

END;



DECLARE

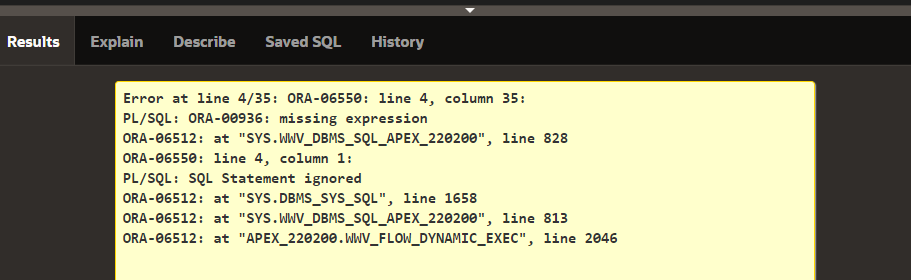
date DATE;

BEGIN

SELECT ADD\_MONTHS(SYSDATE,3) INTO date

FROM dual;

END;



DECLARE

-- converts monthly salary to annual salary

v\_montly\_sal NUMBER(9,2);

v\_annual\_sal NUMBER(9,2);

BEGIN -- begin executable section

…

/\* Compute the annual salary based on the

monthly salary input from the user \*/

v\_annual\_sal := v\_monthly\_sal \* 12;

END; -- end block

DECLARE

v\_outer\_variable VARCHAR2(20):='GLOBAL VARIABLE';

BEGIN

DECLARE

v\_inner\_variable VARCHAR2(20):='LOCAL VARIABLE';

BEGIN

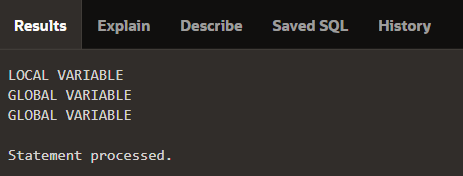
DBMS\_OUTPUT.PUT\_LINE(v\_inner\_variable);

DBMS\_OUTPUT.PUT\_LINE(v\_outer\_variable);

END;

DBMS\_OUTPUT.PUT\_LINE(v\_outer\_variable);

END;



DECLARE

v\_outer\_variable VARCHAR2(20):='GLOBAL VARIABLE';

BEGIN

DECLARE

v\_inner\_variable VARCHAR2(20):='LOCAL VARIABLE';

BEGIN

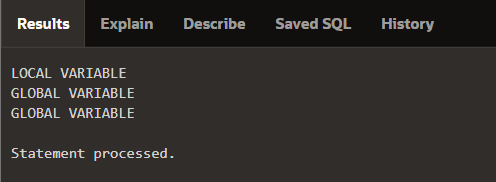
DBMS\_OUTPUT.PUT\_LINE(v\_inner\_variable);

DBMS\_OUTPUT.PUT\_LINE(v\_outer\_variable);

END;

DBMS\_OUTPUT.PUT\_LINE(v\_outer\_variable);

END;



DECLARE

v\_father\_name VARCHAR2(20):='Patrick';

v\_date\_of\_birth DATE:='20-Apr-1972';

BEGIN

DECLARE

v\_child\_name VARCHAR2(20):='Mike';

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Father''s Name: '||v\_father\_name);

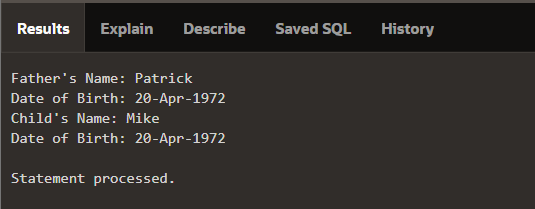
DBMS\_OUTPUT.PUT\_LINE('Date of Birth: '||v\_date\_of\_birth);

DBMS\_OUTPUT.PUT\_LINE('Child''s Name: '||v\_child\_name);

END;

DBMS\_OUTPUT.PUT\_LINE('Date of Birth: '||v\_date\_of\_birth);

END;



DECLARE

v\_outer\_variable VARCHAR2(20):='GLOBAL VARIABLE';

BEGIN

DECLARE

v\_inner\_variable VARCHAR2(20):='LOCAL VARIABLE';

BEGIN

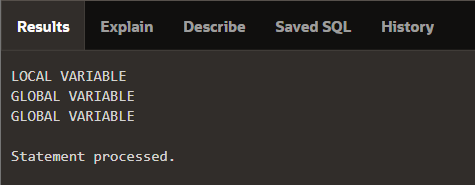
DBMS\_OUTPUT.PUT\_LINE(v\_inner\_variable);

DBMS\_OUTPUT.PUT\_LINE(v\_outer\_variable);

END;

DBMS\_OUTPUT.PUT\_LINE(v\_outer\_variable);

END;



DECLARE

v\_first\_name VARCHAR2(20);

v\_last\_name VARCHAR2(20);

BEGIN

BEGIN

v\_first\_name := 'Carmen';

v\_last\_name := 'Miranda';

DBMS\_OUTPUT.PUT\_LINE

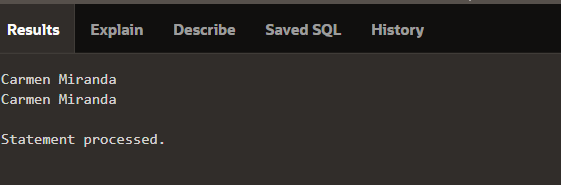
(v\_first\_name || ' ' || v\_last\_name);

END;

DBMS\_OUTPUT.PUT\_LINE

(v\_first\_name || ' ' || v\_last\_name);

END;



DECLARE

v\_first\_name VARCHAR2(20);

BEGIN

DECLARE

v\_last\_name VARCHAR2(20);

BEGIN

v\_first\_name := 'Carmen';

v\_last\_name := 'Miranda';

DBMS\_OUTPUT.PUT\_LINE

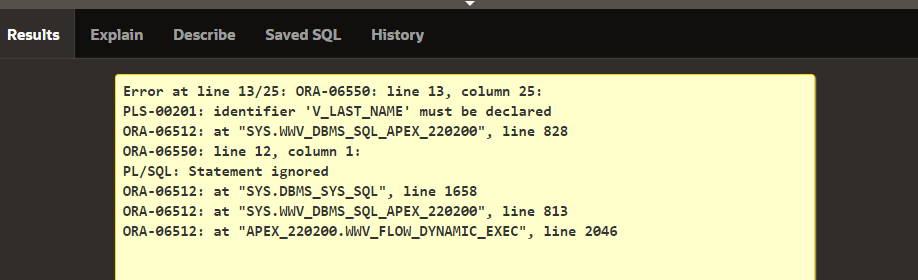
(v\_first\_name || ' ' || v\_last\_name);

END;

DBMS\_OUTPUT.PUT\_LINE

(v\_first\_name || ' ' || v\_last\_name);

END;



DECLARE -- outer block

v\_outervar VARCHAR2(20);

BEGIN

DECLARE -- middle block

v\_middlevar VARCHAR2(20);

BEGIN

BEGIN -- inner block

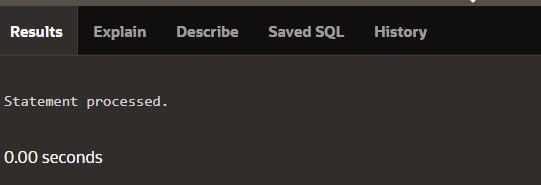
v\_outervar := 'Joachim';

v\_middlevar := 'Chang';

END;

END;

END;



DECLARE

v\_father\_name VARCHAR2(20):='Patrick';

v\_date\_of\_birth DATE:='20-Apr-1972';

BEGIN

DECLARE

v\_child\_name VARCHAR2(20):='Mike';

v\_date\_of\_birth DATE:='12-Dec-2002';

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Father''s Name: ' || v\_father\_name);

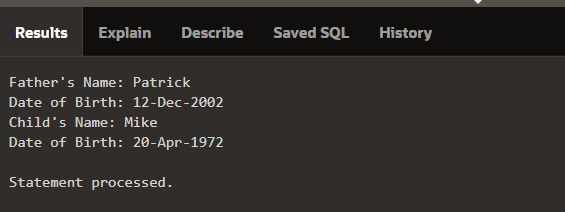
DBMS\_OUTPUT.PUT\_LINE('Date of Birth: ' || v\_date\_of\_birth);

DBMS\_OUTPUT.PUT\_LINE('Child''s Name: ' || v\_child\_name);

END;

DBMS\_OUTPUT.PUT\_LINE('Date of Birth: ' || v\_date\_of\_birth);

END;



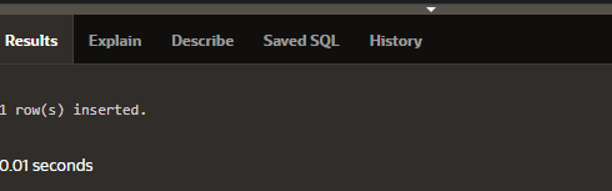
**SECTION 3**

INSERT INTO employees (employee\_id, first\_name,

last\_name, email, hire\_date, job\_id)

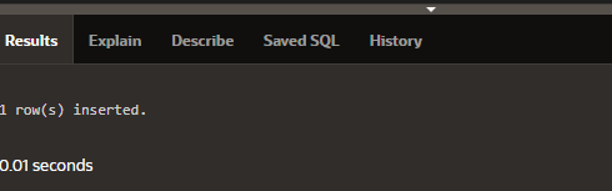
VALUES (305, 'Kareem', 'Naser',

'naserk@oracle.com', SYSDATE, 'SR\_SA\_REP');



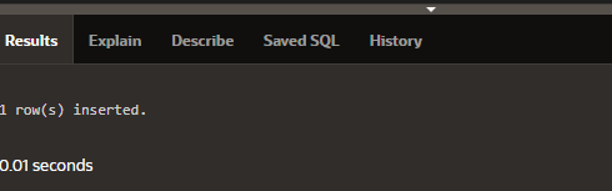
INSERT INTO employees (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_head)

VALUES (305, 'Kareem', 'Naser', 'naserk@oracle.com', '111-222-3333',SYSDATE,'SR\_SA\_REP',7000,NULL,NULL,NULL);



INSERT INTO employees

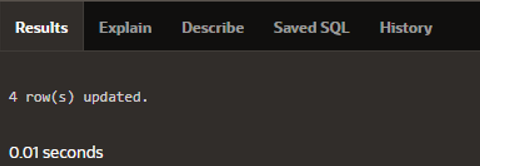
VALUES (305, 'Kareem', 'Naser', 'naserk@oracle.com', '111-222-3333', SYSDATE, 'SR\_SA\_REP', 7000, NULL, NULL, NULL, NULL);



UPDATE employees

SET salary = 11000

WHERE employee\_id = 176;

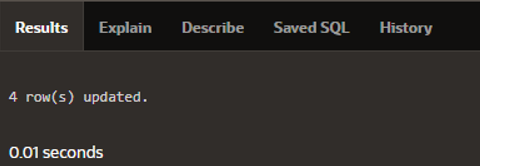


UPDATE employees

SET salary = 11000,

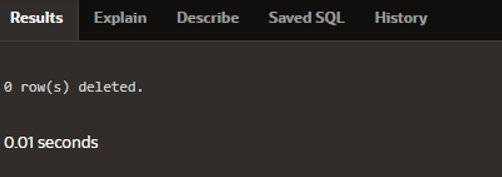
commission\_pct = .3

WHERE employee\_id = 176;



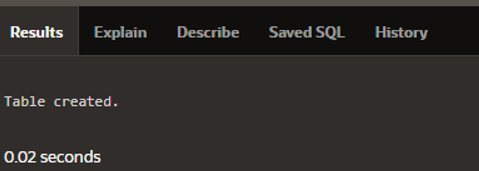
DELETE FROM employees

WHERE employee\_id = 149;



CREATE TABLE bonuses

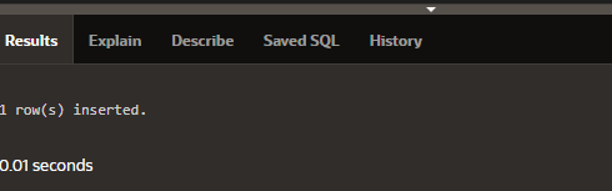
(employee\_id NUMBER(6,0) NOT NULL, bonus NUMBER(8,2) DEFAULT 0);



INSERT INTO bonuses(employee\_id)

(SELECT employee\_id FROM employees

WHERE salary < 10000);



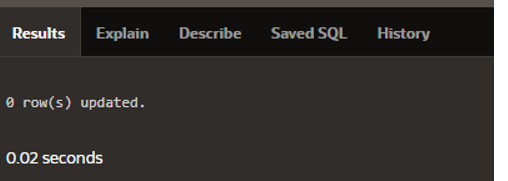
MERGE INTO bonuses b

USING employees e

ON (b.employee\_id = e.employee\_id)

WHEN MATCHED THEN

UPDATE SET b.bonus = e.salary \* .05;



DECLARE

v\_emp\_lname employees.last\_name%TYPE;

BEGIN

SELECT last\_name

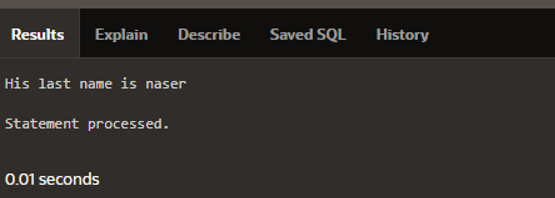
INTO v\_emp\_lname

FROM employees

WHERE employee\_id = 100;

DBMS\_OUTPUT.PUT\_LINE('His last name is ' || v\_emp\_lname);

END;



DECLARE

v\_emp\_hiredate employees.hire\_date%TYPE;

v\_emp\_salary employees.salary%TYPE;

BEGIN

SELECT hire\_date, salary

INTO v\_emp\_hiredate, v\_emp\_salary

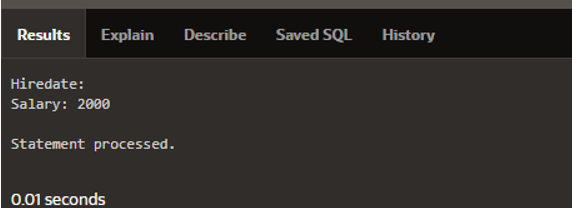
FROM employees

WHERE employee\_id = 100;

DBMS\_OUTPUT.PUT\_LINE('Hiredate: ' || v\_emp\_hiredate);

DBMS\_OUTPUT.PUT\_LINE('Salary: '|| v\_emp\_salary);

END;



DECLARE

v\_salary employees.salary%TYPE;

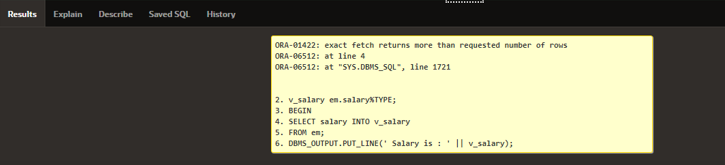
BEGIN

SELECT salary INTO v\_salary

FROM employees;

DBMS\_OUTPUT.PUT\_LINE(' Salary is : ' || v\_salary);

END;



DECLARE

v\_sum\_sal NUMBER(10,2);

v\_deptno NUMBER NOT NULL := 60;

BEGIN

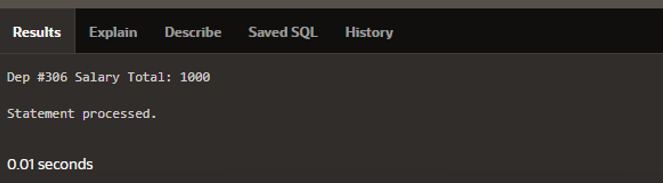
SELECT SUM(salary) -- group function

INTO v\_sum\_sal FROM employees

WHERE department\_id = v\_deptno;

DBMS\_OUTPUT.PUT\_LINE('Dep #60 Salary Total: ' || v\_sum\_sal);

END;



DECLARE

v\_hire\_date employees.hire\_date%TYPE;

employee\_id employees.employee\_id%TYPE := 176;

BEGIN

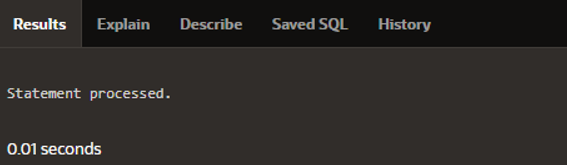
SELECT hire\_date

INTO v\_hire\_date

FROM employees

WHERE employee\_id = employee\_id;

END;



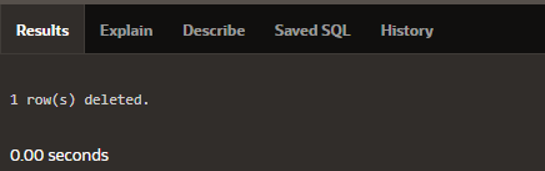
DECLARE

last\_name employees.last\_name%TYPE := 'King';

BEGIN

DELETE FROM emp\_dup WHERE last\_name = last\_name;

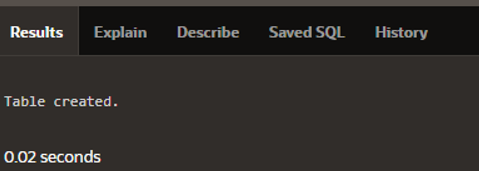
END;



CREATE TABLE copy\_emp

AS SELECT \*

FROM employees;



BEGIN

INSERT INTO copy\_emp

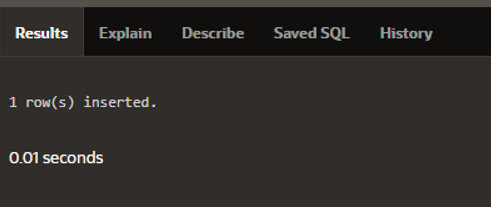
(employee\_id, first\_name, last\_name, email,

hire\_date, job\_id, salary)

VALUES (99, 'Ruth', 'Cores','RCORES', SYSDATE, 'AD\_ASST',

4000);

END;



DECLARE

v\_sal\_increase employees.salary%TYPE := 800;

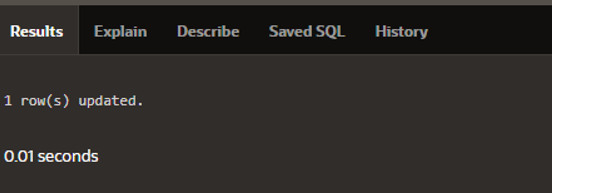
BEGIN

UPDATE copy\_emp

SET salary = salary + v\_sal\_increase

WHERE job\_id = 'ST\_CLERK';

END;



DECLARE

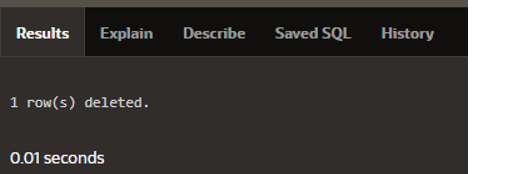
v\_deptno employees.department\_id%TYPE := 10;

BEGIN

DELETE FROM copy\_emp

WHERE department\_id = v\_deptno;

END;



DECLARE

v\_deptno copy\_emp.department\_id%TYPE := 50;

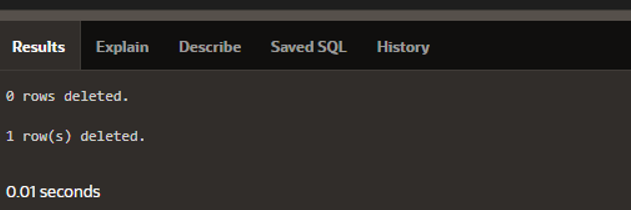
BEGIN

DELETE FROM copy\_emp

WHERE department\_id = v\_deptno;

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || ' rows deleted.');

END;



DECLARE

v\_sal\_increase employees.salary%TYPE := 800;

BEGIN

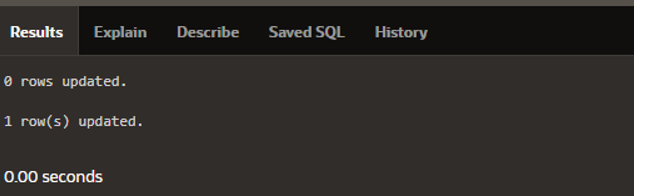
UPDATE copy\_emp

SET salary = salary + v\_sal\_increase

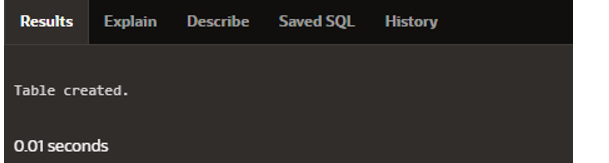
WHERE job\_id = 'ST\_CLERK';

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || ' rows updated.');

END;



CREATE TABLE results (num\_rows NUMBER(4));



DECLARE

v\_rowcount INTEGER;

BEGIN

UPDATE copy\_emp

SET salary = salary + 100

WHERE job\_id = 'ST\_CLERK';

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || ' rows in COPY\_EMPupdated.');

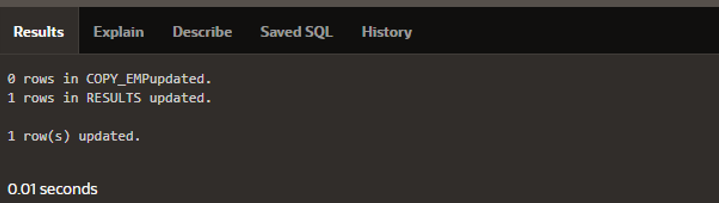
v\_rowcount := SQL%ROWCOUNT;

INSERT INTO results (num\_rows)

VALUES (v\_rowcount);

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || ' rows in RESULTS updated.');

END;



**SECTION 4**

DECLARE

v\_myage NUMBER := 31;

BEGIN

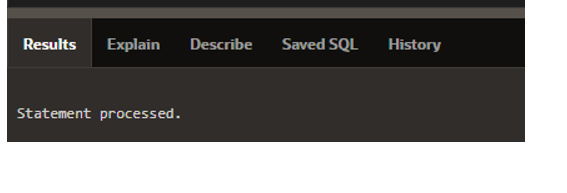
IF v\_myage < 11

THEN

DBMS\_OUTPUT.PUT\_LINE('I am a child');

END IF;

END;



DECLARE

v\_myage NUMBER := 31;

BEGIN

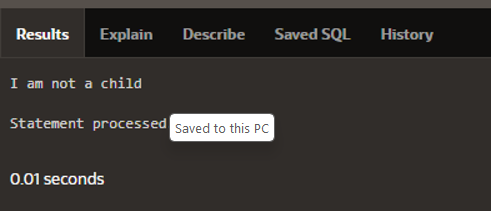
IF v\_myage < 11

THEN

DBMS\_OUTPUT.PUT\_LINE('I am a child');

END IF;

END;



DECLARE

v\_myage NUMBER := 31;

BEGIN

IF v\_myage < 11

THEN

DBMS\_OUTPUT.PUT\_LINE('I am a child');

ELSIF v\_myage < 20

THEN

DBMS\_OUTPUT.PUT\_LINE('I am young');

ELSIF v\_myage < 30

THEN

DBMS\_OUTPUT.PUT\_LINE('I am in my twenties');

ELSIF v\_myage < 40

THEN

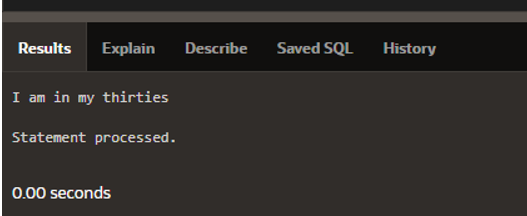
DBMS\_OUTPUT.PUT\_LINE('I am in my thirties');

ELSE

DBMS\_OUTPUT.PUT\_LINE('I am mature');

END IF;

END;



DECLARE

v\_myage NUMBER := 31;

v\_myfirstname VARCHAR2(11) := 'Christopher';

BEGIN

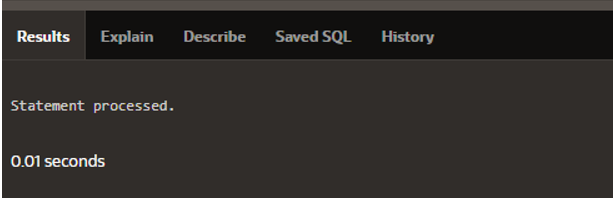
IF v\_myfirstname ='Christopher' AND v\_myage < 11

THEN

DBMS\_OUTPUT.PUT\_LINE('I am a child named Christopher');

END IF;

END;



DECLARE

v\_myage NUMBER := 31;

v\_myfirstname VARCHAR2(11) := 'Christopher';

BEGIN

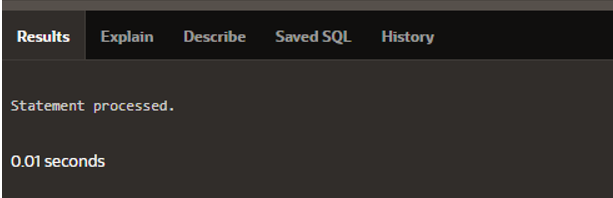
IF v\_myfirstname ='Christopher' AND v\_myage < 11

THEN

DBMS\_OUTPUT.PUT\_LINE('I am a child named Christopher');

END IF;

END;



DECLARE

v\_myage NUMBER;

BEGIN

IF v\_myage < 11

THEN

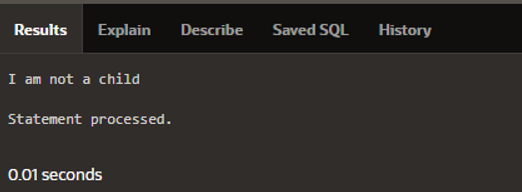
DBMS\_OUTPUT.PUT\_LINE('I am a child');

ELSE

DBMS\_OUTPUT.PUT\_LINE('I am not a child');

END IF;

END;



DECLARE

v\_num NUMBER := 15;

v\_txt VARCHAR2(50);

BEGIN

CASE v\_num

WHEN 20 THEN v\_txt := 'number equals 20';

WHEN 17 THEN v\_txt := 'number equals 17';

WHEN 15 THEN v\_txt := 'number equals 15';

WHEN 13 THEN v\_txt := 'number equals 13';

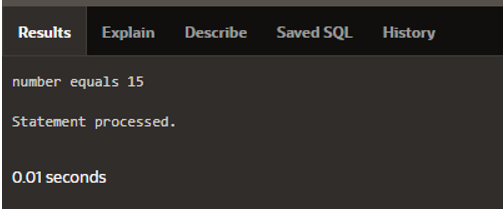
WHEN 10 THEN v\_txt := 'number equals 10';

ELSE v\_txt := 'some other number';

END CASE;

DBMS\_OUTPUT.PUT\_LINE(v\_txt);

END;



DECLARE

v\_num NUMBER := 15;

v\_txt VARCHAR2(50);

BEGIN

CASE

WHEN v\_num > 20 THEN v\_txt := 'greater than 20';

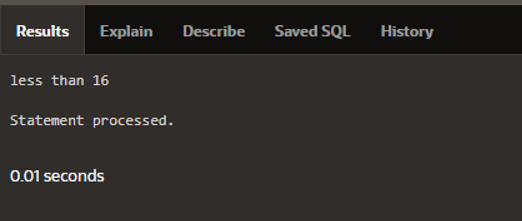
WHEN v\_num > 15 THEN v\_txt := 'greater than 15';

ELSE v\_txt := 'less than 16';

END CASE;

DBMS\_OUTPUT.PUT\_LINE(v\_txt);

END;



DECLARE

v\_grade CHAR(1) := 'A';

v\_appraisal VARCHAR2(20);

BEGIN

v\_appraisal :=

CASE v\_grade

WHEN 'A' THEN 'Excellent'

WHEN 'B' THEN 'Very Good'

WHEN 'C' THEN 'Good'

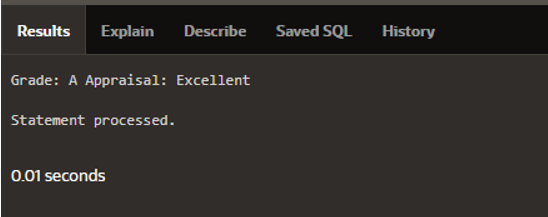
ELSE 'No such grade'

END;

DBMS\_OUTPUT.PUT\_LINE('Grade: ' || v\_grade ||

' Appraisal: ' || v\_appraisal);

END;



DECLARE

v\_out\_var VARCHAR2(15);

v\_in\_var NUMBER := 20;

BEGIN

v\_out\_var :=

CASE v\_in\_var

WHEN 1 THEN 'Low value'

WHEN v\_in\_var THEN 'Same value'

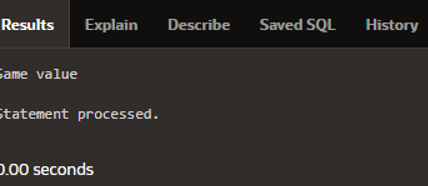
WHEN 20 THEN 'Middle value'

ELSE 'Other value'

END;

DBMS\_OUTPUT.PUT\_LINE(v\_out\_var);

END;



DECLARE

v\_grade CHAR(1) := 'A';

v\_appraisal VARCHAR2(20);

BEGIN

v\_appraisal :=

CASE -- no selector here

WHEN v\_grade = 'A' THEN 'Excellent'

WHEN v\_grade IN ('B','C') THEN 'Good'

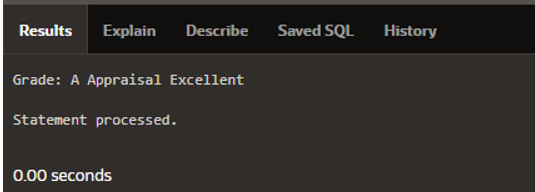
ELSE 'No such grade'

END;

DBMS\_OUTPUT.PUT\_LINE ('Grade: '|| v\_grade ||

' Appraisal ' || v\_appraisal);

END;



DECLARE

v\_grade CHAR(1) := 'A';

v\_appraisal VARCHAR2(20);

BEGIN

v\_appraisal :=

CASE

WHEN v\_grade = 'A' THEN 'Excellent'

WHEN v\_grade IN ('B','C') THEN 'Good'

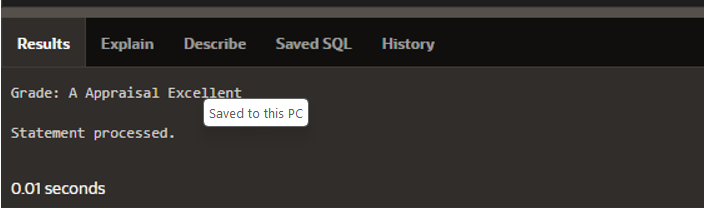
ELSE 'No such grade'

END;

DBMS\_OUTPUT.PUT\_LINE ('Grade: '|| v\_grade || ' Appraisal '

|| v\_appraisal);

END;



DECLARE

v\_grade CHAR(1) := 'A';

BEGIN

CASE

WHEN v\_grade = 'A' THEN

DBMS\_OUTPUT.PUT\_LINE ('Excellent');

WHEN v\_grade IN ('B','C') THEN

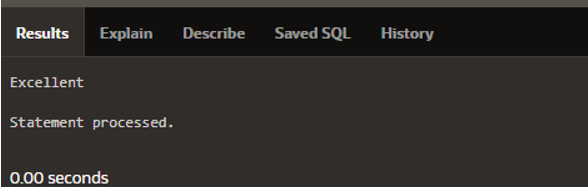
DBMS\_OUTPUT.PUT\_LINE ('Good');

ELSE

DBMS\_OUTPUT.PUT\_LINE('No such grade');

END CASE;

END;



DECLARE

v\_counter NUMBER(2) := 1;

BEGIN

LOOP

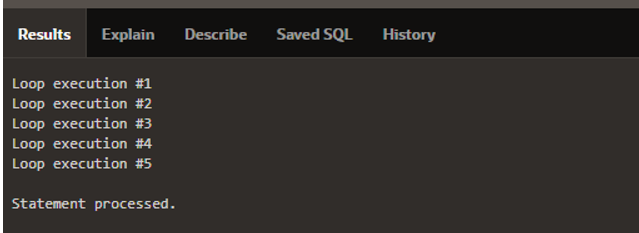
DBMS\_OUTPUT.PUT\_LINE('Loop execution #' || v\_counter);

v\_counter := v\_counter + 1;

EXIT WHEN v\_counter > 5;

END LOOP;

END;



DECLARE

v\_counter NUMBER := 1;

BEGIN

LOOP

DBMS\_OUTPUT.PUT\_LINE('Counter is ' || v\_counter);

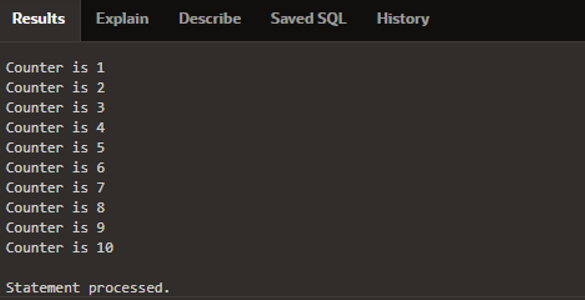
v\_counter := v\_counter + 1;

IF v\_counter > 10 THEN EXIT;

END IF;

END LOOP;

END;



DECLARE

v\_counter NUMBER := 1;

BEGIN

LOOP

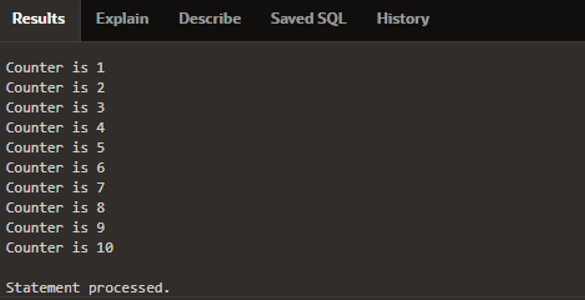
DBMS\_OUTPUT.PUT\_LINE('Counter is ' || v\_counter);

v\_counter := v\_counter + 1;

EXIT WHEN v\_counter > 10;

END LOOP;

END;



DECLARE

v\_loc\_id locations.location\_id%TYPE;

v\_counter NUMBER := 1;

BEGIN

SELECT MAX(location\_id) INTO v\_loc\_id FROM locations

WHERE country\_id = 2;

WHILE v\_counter <= 3 LOOP

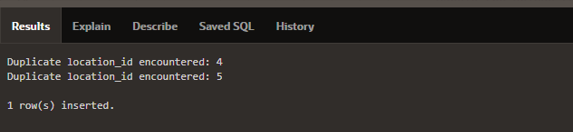
INSERT INTO locations(location\_id, city, country\_id)

VALUES((v\_loc\_id + v\_counter), 'Montreal', 2);

v\_counter := v\_counter + 1;

END LOOP;

END;



DECLARE

v\_loc\_id locations.location\_id%TYPE;

v\_counter NUMBER := 1;

BEGIN

SELECT MAX(location\_id) INTO v\_loc\_id FROM locations

WHERE country\_id = 2;

WHILE v\_counter <= 3 LOOP

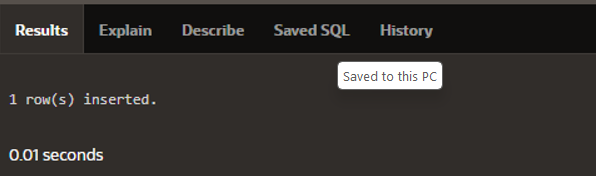
INSERT INTO locations(location\_id, city, country\_id)

VALUES((v\_loc\_id + v\_counter), 'Montreal', 2);

v\_counter := v\_counter + 1;

END LOOP;

END;



DECLARE

v\_loc\_id locations.location\_id%TYPE;

v\_counter NUMBER := 1;

BEGIN

SELECT MAX(location\_id) INTO v\_loc\_id FROM locations

WHERE country\_id = 2;

WHILE v\_counter <= 3 LOOP

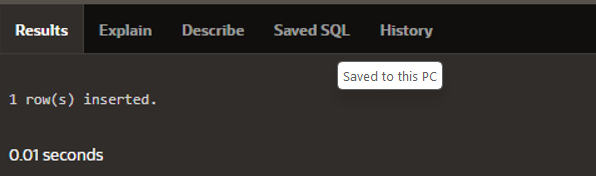
INSERT INTO locations(location\_id, city, country\_id)

VALUES((v\_loc\_id + v\_counter), 'Montreal', 2);

v\_counter := v\_counter + 1;

END LOOP;

END;



DECLARE

v\_loc\_id locations.location\_id%TYPE;

BEGIN

SELECT MAX(location\_id) INTO v\_loc\_id FROM locations

WHERE country\_id = 2;

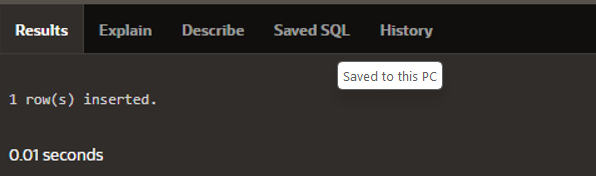
FOR i IN 1..3 LOOP

INSERT INTO locations(location\_id, city, country\_id)

VALUES((v\_loc\_id + i), 'Montreal', 2);

END LOOP;

END;



BEGIN

FOR v\_outerloop IN 1..3 LOOP

FOR v\_innerloop IN REVERSE 1..5 LOOP

DBMS\_OUTPUT.PUT\_LINE('Outer loop is: ' ||

v\_outerloop ||

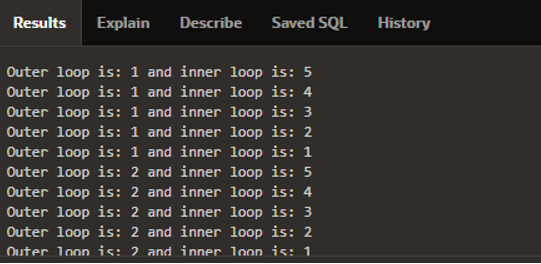
' and inner loop is: ' ||

v\_innerloop);

END LOOP;

END LOOP;

END;



DECLARE

v\_outerloop PLS\_INTEGER := 0;

v\_innerloop PLS\_INTEGER := 5;

BEGIN

<<outer\_loop>>

LOOP

v\_outerloop := v\_outerloop + 1;

v\_innerloop := 5;

EXIT WHEN v\_outerloop > 3;

<<inner\_loop>>

LOOP

DBMS\_OUTPUT.PUT\_LINE('Outer loop is: ' || v\_outerloop ||

' and inner loop is: ' || v\_innerloop);

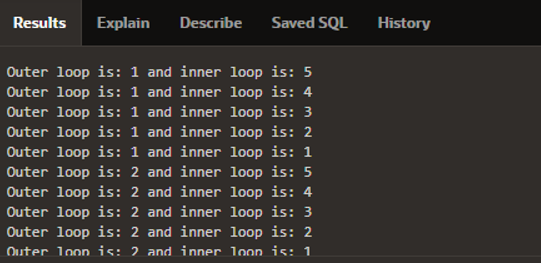
v\_innerloop := v\_innerloop - 1;

EXIT WHEN v\_innerloop = 0;

END LOOP inner\_loop;

END LOOP outer\_loop;

END;



**SECTION 5**

WITH avg\_speed AS (

SELECT RoadID, AVG(Speed) AS AvgSpeed

FROM TrafficData

GROUP BY RoadID

)

SELECT RoadID, RoadName, AvgSpeed

FROM (

SELECT Roads.RoadID, Roads.RoadName, avg\_speed.AvgSpeed, ROWNUM AS r

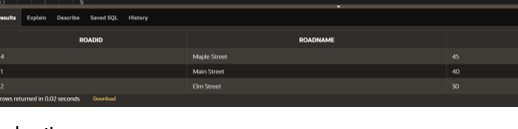
FROM avg\_speed

JOIN Roads ON avg\_speed.RoadID = Roads.RoadID

ORDER BY avg\_speed.AvgSpeed DESC

)

WHERE r <= 3;



SELECT

CategoryID,

CategoryName,

LEVEL AS HierarchyLevel,

SYS\_CONNECT\_BY\_PATH(CategoryName, ' > ') AS Path

FROM

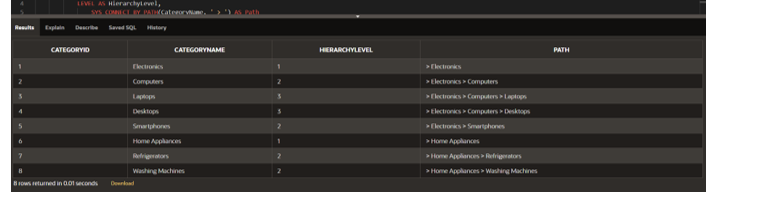
Categories

START WITH

ParentCategoryID IS NULL

CONNECT BY

PRIOR CategoryID = ParentCategoryID;



SELECT

TO\_CHAR(TransactionDate, 'YYYY-MM') AS Month,

COUNT(DISTINCT CustomerID) AS TotalDistinctCustomers

FROM

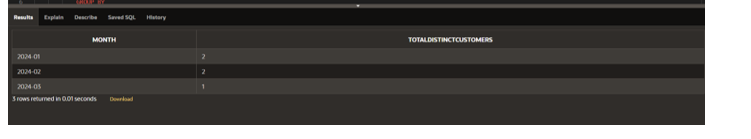
CustomerTransactions

GROUP BY

TO\_CHAR(TransactionDate, 'YYYY-MM')

ORDER BY

Month;



WITH DistanceCalculation AS (

SELECT

i1.IntersectionID AS IntersectionID1,

i1.IntersectionName AS IntersectionName1,

i2.IntersectionID AS IntersectionID2,

i2.IntersectionName AS IntersectionName2,

(6371 \* ACOS(

COS((i1.Latitude \* 3.141592653589793 / 180)) \* COS((i2.Latitude \* 3.141592653589793 / 180)) \* COS((i2.Longitude \* 3.141592653589793 / 180) - (i1.Longitude \* 3.141592653589793 / 180)) +

SIN((i1.Latitude \* 3.141592653589793 / 180)) \* SIN((i2.Latitude \* 3.141592653589793 / 180))

)) AS Distance

FROM

Intersections i1

JOIN

Intersections i2

ON

i1.IntersectionID <> i2.IntersectionID

)

SELECT

IntersectionID1,

IntersectionName1,

IntersectionID2,

IntersectionName2,

Distance

FROM

DistanceCalculation

ORDER BY

Distance



SELECT

o.OrderID,

o.OrderDate,

o.CustomerID,

o.TotalAmount

FROM

Orders o

WHERE

o.OrderDate >= DATE\_SUB(CURDATE(), INTERVAL 7 DAY)

ORDER BY

o.OrderDate DESC;

CREATE OR REPLACE FUNCTION safe\_divide(

p\_numerator NUMBER,

p\_denominator NUMBER

) RETURN NUMBER IS

BEGIN

IF p\_denominator = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Division by zero is not allowed');

END IF;

RETURN p\_numerator / p\_denominator;

END safe\_divide;

CREATE OR REPLACE PROCEDURE update\_rows(

p\_ids IN SYS.ODCINUMBERLIST,

p\_new\_values IN SYS.ODCIVARCHAR2LIST

) IS

BEGIN

FORALL i IN 1..p\_ids.COUNT

UPDATE my\_table

SET column\_name = p\_new\_values(i)

WHERE id = p\_ids(i);

END update\_rows;

**Solution**

CREATE OR REPLACE PROCEDURE insert\_into\_nested\_table(

p\_data IN SYS.ODCINUMBERLIST

) IS

v\_nested\_table my\_table\_type := my\_table\_type();

BEGIN

FOR i IN 1..p\_data.COUNT

v\_nested\_table.EXTEND;

v\_nested\_table(i) := p\_data(i);

END LOOP;

INSERT INTO my\_table (nested\_column)

VALUES (v\_nested\_table);

END insert\_into\_nested\_table;

CREATE OR REPLACE PROCEDURE execute\_dynamic\_sql(

p\_sql IN VARCHAR2

) IS

v\_cursor SYS\_REFCURSOR;

v\_row my\_table%ROWTYPE;

BEGIN

OPEN v\_cursor FOR p\_sql;

LOOP

FETCH v\_cursor INTO v\_row;

EXIT WHEN v\_cursor%NOTFOUND;

-- Process the row data

DBMS\_OUTPUT.PUT\_LINE(v\_row.column\_name);

END LOOP;

CLOSE v\_cursor;

END execute\_dynamic\_sql;

CREATE OR REPLACE PACKAGE sales\_data\_pkg IS

TYPE sales\_data\_type IS RECORD (

sale\_date DATE,

product\_name VARCHAR2(50),

quantity NUMBER,

total\_amount NUMBER

);

TYPE sales\_data\_table IS TABLE OF sales\_data\_type;

FUNCTION get\_sales\_data(

p\_start\_date DATE,

p\_end\_date DATE

) RETURN sales\_data\_table PIPELINED;

END sales\_data\_pkg;

CREATE OR REPLACE PACKAGE BODY sales\_data\_pkg IS

FUNCTION get\_sales\_data(

p\_start\_date DATE,

p\_end\_date DATE

) RETURN sales\_data\_table PIPELINED IS

v\_sale\_date DATE;

v\_product\_name VARCHAR2(50);

v\_quantity NUMBER;

v\_total\_amount NUMBER;

CURSOR sales\_cur IS

SELECT sale\_date, product\_name, quantity, total\_amount

FROM sales

WHERE sale\_date BETWEEN p\_start\_date AND p\_end\_date;

BEGIN

FOR sales\_rec IN sales\_cur LOOP

v\_sale\_date := sales\_rec.sale\_date;

v\_product\_name := sales\_rec.product\_name;

v\_quantity := sales\_rec.quantity; -- Assign the quantity value

v\_total\_amount := sales\_rec.total\_amount;

PIPE ROW(sales\_data\_type(v\_sale\_date, v\_product\_name, v\_quantity, v\_total\_amount));

END LOOP;

CLOSE sales\_cur;

RETURN;

END get\_sales\_data;

END sales\_data\_pkg;

**SECTION 6**

DECLARE

v\_employee\_id employees.employee\_id%TYPE;

v\_first\_name employees.first\_name%TYPE;

v\_last\_name employees.last\_name%TYPE;

v\_email employees.email%TYPE;

... FIVE MORE SCALAR VARIABLES REQUIRED TO MATCH THE TABLE

v\_manager\_id employees.manager\_id%TYPE;

v\_department\_id employees.department\_id%TYPE;

BEGIN

SELECT employee\_id, first\_name, ... EIGHT MORE HERE,

department\_id

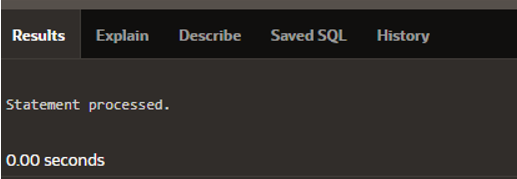
INTO v\_employee\_id, v\_first\_name, ... AND HERE,

v\_department\_id

FROM employees

WHERE employee\_id = 100;

END;



DECLARE

v\_emp\_record employees%ROWTYPE;

BEGIN

SELECT \* INTO v\_emp\_record

FROM employees

WHERE employee\_id = 100;

DBMS\_OUTPUT.PUT\_LINE('Email for ' || v\_emp\_record.first\_name

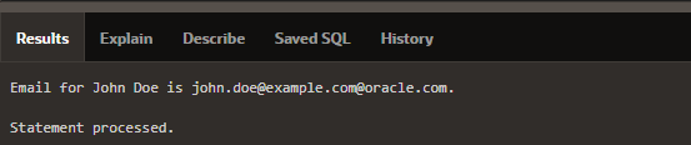
||

' ' || v\_emp\_record.last\_name || ' is ' ||

v\_emp\_record.email ||

'@oracle.com.');

END;



DECLARE

v\_emp\_record employees%ROWTYPE;

v\_emp\_copy\_record v\_emp\_record%ROWTYPE;

BEGIN

SELECT \* INTO v\_emp\_record

FROM employees

WHERE employee\_id = 100;

v\_emp\_copy\_record := v\_emp\_record;

v\_emp\_copy\_record.salary := v\_emp\_record.salary \* 1.2;

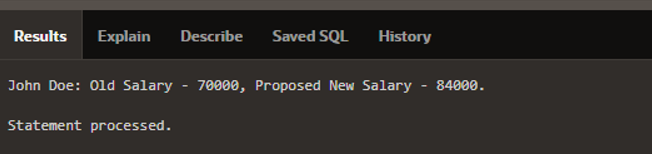
DBMS\_OUTPUT.PUT\_LINE(v\_emp\_record.first\_name ||

' ' || v\_emp\_record.last\_name || ': Old Salary - ' ||

v\_emp\_record.salary || ', Proposed New Salary - ' ||

v\_emp\_copy\_record.salary || '.');

END;



DECLARE

TYPE person\_dept IS RECORD

(first\_name employees.first\_name%TYPE,

last\_name employees.last\_name%TYPE,

department\_name departments.department\_name%TYPE);

v\_person\_dept\_rec person\_dept;

BEGIN

SELECT e.first\_name, e.last\_name, d.department\_name

INTO v\_person\_dept\_rec

FROM employees e JOIN departments d

ON e.department\_id = d.department\_id

WHERE employee\_id = 200;

DBMS\_OUTPUT.PUT\_LINE(v\_person\_dept\_rec.first\_name ||

' ' || v\_person\_dept\_rec.last\_name || ' is in the ' ||

v\_person\_dept\_rec.department\_name || ' department.');

END;

DECLARE -- outer block

TYPE employee\_type IS RECORD

(first\_name employees.first\_name%TYPE := 'Amy');

v\_emp\_rec\_outer employee\_type;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(v\_emp\_rec\_outer.first\_name);

DECLARE -- inner block

v\_emp\_rec\_inner employee\_type;

BEGIN

v\_emp\_rec\_outer.first\_name := 'Clara';

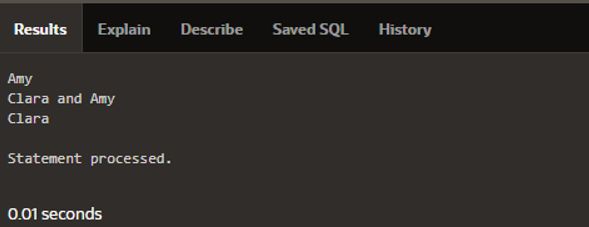
DBMS\_OUTPUT.PUT\_LINE(v\_emp\_rec\_outer.first\_name ||

' and ' || v\_emp\_rec\_inner.first\_name);

END;

DBMS\_OUTPUT.PUT\_LINE(v\_emp\_rec\_outer.first\_name);

END;



DECLARE

TYPE t\_hire\_date IS TABLE OF employees.hire\_date%TYPE

INDEX BY BINARY\_INTEGER;

v\_hire\_date\_tab t\_hire\_date;

BEGIN

FOR emp\_rec IN

(SELECT employee\_id, hire\_date FROM employees)

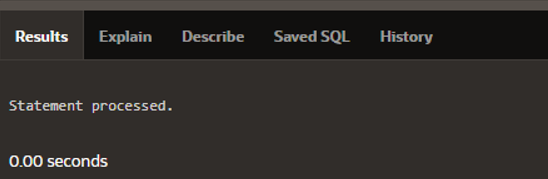
LOOP

v\_hire\_date\_tab(emp\_rec.employee\_id)

:= emp\_rec.hire\_date;

END LOOP;

END;



DECLARE

TYPE t\_hire\_date IS TABLE OF employees.hire\_date%TYPE

INDEX BY BINARY\_INTEGER;

v\_hire\_date\_tab t\_hire\_date;

v\_count BINARY\_INTEGER := 0;

BEGIN

FOR emp\_rec IN

(SELECT hire\_date FROM employees)

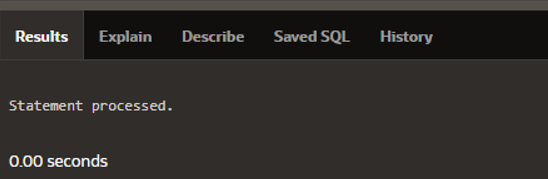
LOOP

v\_count := v\_count + 1;

v\_hire\_date\_tab(v\_count) := emp\_rec.hire\_date;

END LOOP;

END;



DECLARE

TYPE t\_hire\_date IS TABLE OF employees.hire\_date%TYPE

INDEX BY BINARY\_INTEGER;

v\_hire\_date\_tab t\_hire\_date;

v\_hire\_date\_count NUMBER(4);

BEGIN

FOR emp\_rec IN

(SELECT employee\_id, hire\_date FROM employees)

LOOP

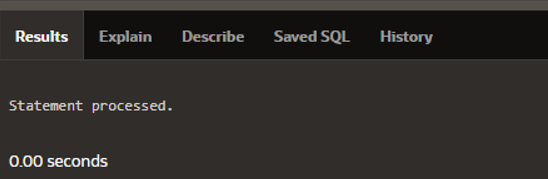
v\_hire\_date\_tab(emp\_rec.employee\_id)

:= emp\_rec.hire\_date;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(v\_hire\_date\_tab.COUNT);

END;



**SECTION 7**

DECLARE

v\_country\_name countries.country\_name%TYPE

:= 'Republic of Korea';

v\_elevation countries.highest\_elevation%TYPE;

BEGIN

SELECT highest\_elevation

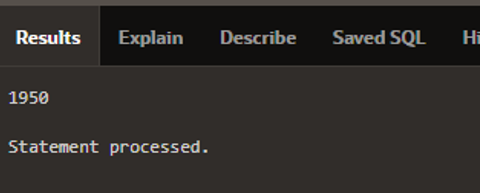
INTO v\_elevation

FROM countries

WHERE country\_name = v\_country\_name;

DBMS\_OUTPUT.PUT\_LINE(v\_elevation);

END;



DECLARE

v\_country\_name countries.country\_name%TYPE := 'Korea, South';

v\_elevation countries.highest\_elevation%TYPE;

BEGIN

SELECT highest\_elevation INTO v\_elevation

FROM countries WHERE country\_name = v\_country\_name;

EXCEPTION

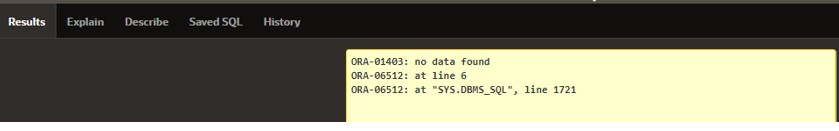
WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE ('Country name, ' || v\_country\_name || ',

cannot be found. Re-enter the country name using the correct

spelling.');

END;



DECLARE

v\_country\_name countries.country\_name%TYPE := 'Korea, South';

v\_elevation countries.highest\_elevation%TYPE;

BEGIN

SELECT highest\_elevation INTO v\_elevation

FROM countries WHERE country\_name = v\_country\_name;

DBMS\_OUTPUT.PUT\_LINE(v\_elevation); -- Point A

EXCEPTION

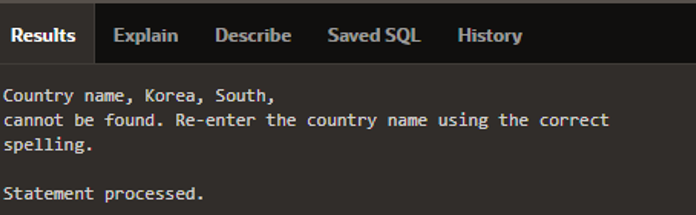
WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE ('Country name, ' || v\_country\_name || ',

cannot be found. Re-enter the country name using the correct

spelling.');

END;



DECLARE

v\_lname VARCHAR2(15);

BEGIN

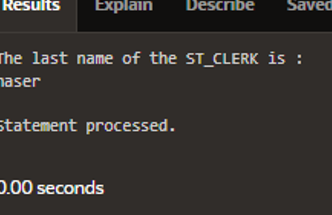
SELECT last\_name INTO v\_lname

FROM employees WHERE job\_id = 'ST\_CLERK';

DBMS\_OUTPUT.PUT\_LINE('The last name of the ST\_CLERK is :

'||v\_lname);

END;



DECLARE

v\_lname employees.last\_name%TYPE;

BEGIN

SELECT last\_name INTO v\_lname

FROM employees WHERE job\_id = 'ST\_CLERK';

DBMS\_OUTPUT.PUT\_LINE('The last name of the ST\_CLERK is: '

|| v\_lname);

EXCEPTION

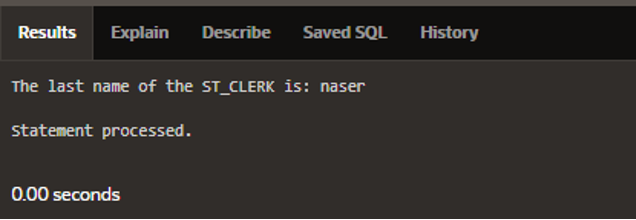
WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE ('Your select statement retrieved

multiple rows. Consider using a

cursor.');

END;



EXCEPTION

WHEN exception1 [OR exception2 . . .] THEN

statement1;

statement2;

. . .

[WHEN exception3 [OR exception4 . . .] THEN

statement1;

statement2;

. . .]

[WHEN OTHERS THEN

statement1;

statement

DECLARE

v\_lname VARCHAR2(15);

BEGIN

SELECT last\_name INTO v\_lname

FROM employees WHERE job\_id = 'ST\_CLERK';

DBMS\_OUTPUT.PUT\_LINE('The last name of the ST\_CLERK is: ' ||

v\_lname);

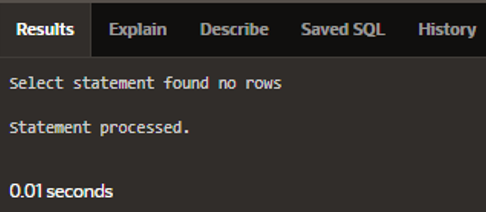
EXCEPTION

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE ('Your select statement retrieved multiple

rows. Consider using a cursor.');

END;



DECLARE

v\_lname VARCHAR2(15);

BEGIN

SELECT last\_name INTO v\_lname

FROM employees WHERE job\_id = 'ST\_CLERK';

DBMS\_OUTPUT.PUT\_LINE('The last name of the ST\_CLERK is:

'||v\_lname);

EXCEPTION

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE ('Select statement found multiple

rows');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE ('Select statement found no rows');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE ('Another type of error occurred');

END;

DECLARE

v\_name VARCHAR2(20):= 'Accounting';

v\_deptno NUMBER := 27;

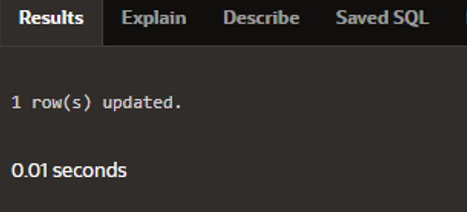
BEGIN

UPDATE departments

SET department\_name = v\_name

WHERE department\_id = v\_deptno;

END;



EXCEPTION

WHEN e\_invalid\_department THEN

DBMS\_OUTPUT.PUT\_LINE('No such department id.');

DECLARE

e\_invalid\_department EXCEPTION;

v\_name VARCHAR2(20):='Accounting';

v\_deptno NUMBER := 27;

BEGIN

UPDATE departments

SET department\_name = v\_name

WHERE department\_id = v\_deptno;

IF SQL%NOTFOUND THEN

RAISE e\_invalid\_department;

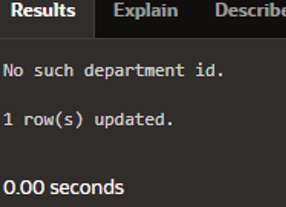
END IF;

EXCEPTION

WHEN e\_invalid\_department

THEN DBMS\_OUTPUT.PUT\_LINE('No such department id.');

END;



DECLARE

v\_mgr PLS\_INTEGER := 123;

BEGIN

DELETE FROM employees

WHERE manager\_id = v\_mgr;

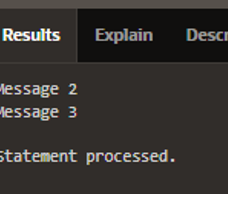
IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20202,

'This is not a valid manager');

END IF;

END;



DECLARE

v\_mgr PLS\_INTEGER := 27;

v\_employee\_id employees.employee\_id%TYPE;

BEGIN

SELECT employee\_id INTO v\_employee\_id

FROM employees

WHERE manager\_id = v\_mgr;

DBMS\_OUTPUT.PUT\_LINE('Employee #' || v\_employee\_id ||

' works for manager #' || v\_mgr || '.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20201,

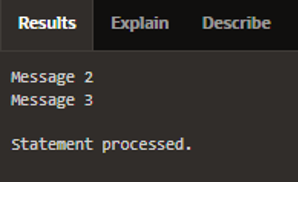
'This manager has no employees');

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20202,

'Too many employees were found.');

END;



DECLARE

v\_last\_name employees.last\_name%TYPE;

BEGIN

BEGIN

SELECT last\_name INTO v\_last\_name

FROM employees WHERE employee\_id = 999;

DBMS\_OUTPUT.PUT\_LINE('Message 1');

EXCEPTION

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Message 2');

END;

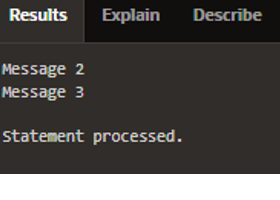
DBMS\_OUTPUT.PUT\_LINE('Message 3');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Message 4');

END;



BEGIN

DECLARE

e\_myexcep EXCEPTION;

BEGIN

RAISE e\_myexcep;

DBMS\_OUTPUT.PUT\_LINE('Message 1');

EXCEPTION

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Message 2');

END;

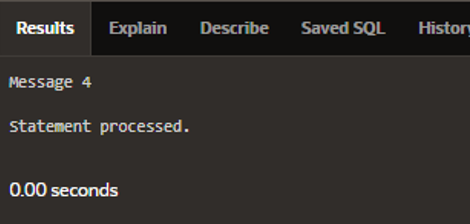
DBMS\_OUTPUT.PUT\_LINE('Message 3');

EXCEPTION

WHEN e\_myexcep THEN

DBMS\_OUTPUT.PUT\_LINE('Message 4');

END;



DECLARE

e\_myexcep EXCEPTION;

BEGIN

BEGIN

RAISE e\_myexcep;

DBMS\_OUTPUT.PUT\_LINE('Message 1');

EXCEPTION

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Message 2');

END;

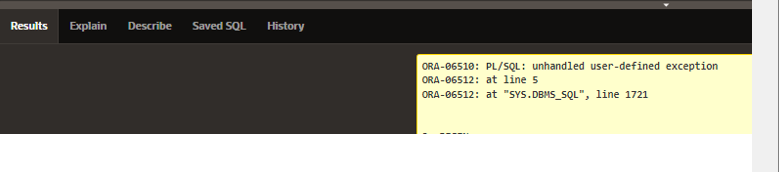
DBMS\_OUTPUT.PUT\_LINE('Message 3');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Message 4');

END;



**SECTION 8**

CREATE OR REPLACE PROCEDURE add\_dept IS

v\_dept\_id dept.department\_id%TYPE;

v\_dept\_name dept.department\_name%TYPE;

BEGIN

v\_dept\_id := 280;

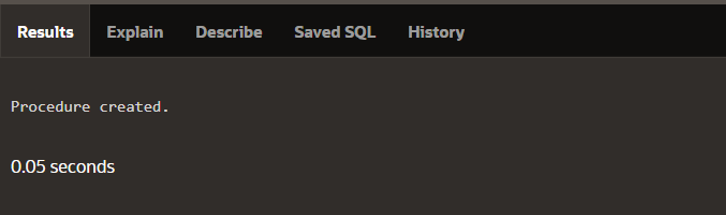
v\_dept\_name := 'ST-Curriculum';

INSERT INTO dept(department\_id, department\_name)

VALUES(v\_dept\_id, v\_dept\_name);

DBMS\_OUTPUT.PUT\_LINE('Inserted '|| SQL%ROWCOUNT || ' row.');

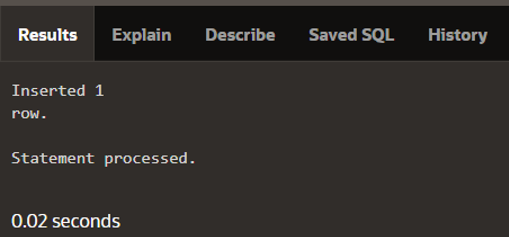
END;



BEGIN

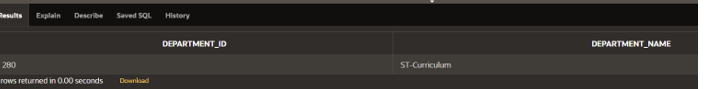
add\_dept;

END;



SELECT department\_id, department\_name FROM dept WHERE

department\_id=280;



CREATE OR REPLACE PROCEDURE delete\_emp

(p\_emp\_id IN employees.employee\_id%TYPE)

IS

PROCEDURE log\_emp (p\_emp IN employees.employee\_id%TYPE)

IS BEGIN

INSERT INTO logging\_table VALUES(p\_emp, ...);

END log\_emp;

BEGIN

DELETE FROM employees

WHERE employee\_id = p\_emp\_id;

log\_emp(p\_emp\_id);

END delete\_emp;

CREATE OR REPLACE PROCEDURE raise\_salary

(p\_id IN my\_employees.employee\_id%TYPE,

p\_percent IN NUMBER)

IS

BEGIN

UPDATE my\_employees

SET salary = salary \* (1 + p\_percent/100)

WHERE employee\_id = p\_id;

END raise\_salary

BEGIN raise\_salary (176, 10); END;

CREATE OR REPLACE PROCEDURE raise\_salary

(p\_id IN my\_employees.employee\_id%TYPE,

p\_percent IN NUMBER)

IS

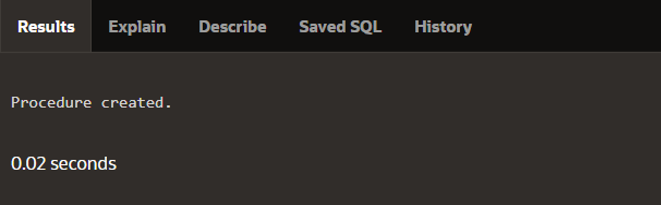
BEGIN

UPDATE my\_employees

SET salary = salary \* (1 + p\_percent/100)

WHERE employee\_id = p\_id;

END raise\_salary;



DECLARE

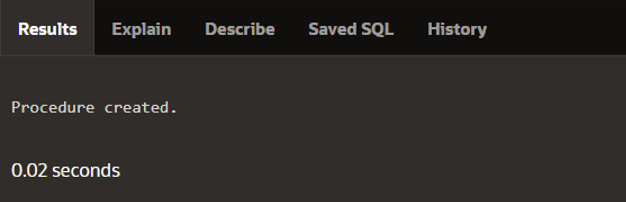
a\_emp\_name employees.last\_name%TYPE;

a\_emp\_sal employees.salary%TYPE;

BEGIN

query\_emp(178, a\_emp\_name, a\_emp\_sal); ...

END;

 DECLARE

a\_phone\_no VARCHAR2(13);

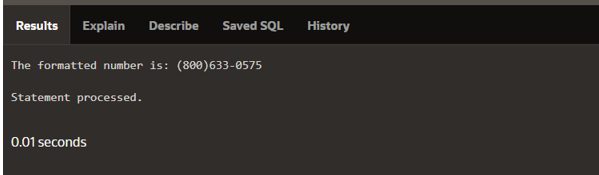
BEGIN

a\_phone\_no := '8006330575' ;

format\_phone(a\_phone\_no);

DBMS\_OUTPUT.PUT\_LINE('The formatted number is: ' || a\_phone\_no);

END;



CREATE OR REPLACE PROCEDURE add\_dept(

p\_name IN my\_depts.department\_name%TYPE,

p\_loc IN my\_depts.location\_id%TYPE) IS

BEGIN

INSERT INTO my\_depts(department\_id,

department\_name, location\_id)

VALUES (departments\_seq.NEXTVAL, p\_name, p\_loc);

END add\_dept;

)

**SECTION 9**

CREATE [OR REPLACE] FUNCTION function\_name

[(parameter1 [mode1] datatype1, ...)]

RETURN datatype IS|AS

[local\_variable\_declarations; …]

BEGIN

-- actions;

RETURN expression;

END [function\_name];

CREATE OR REPLACE FUNCTION get\_sal

(p\_id IN employees.employee\_id%TYPE)

RETURN NUMBER IS

v\_sal employees.salary%TYPE := 0;

BEGIN

SELECT salary

INTO v\_sal

FROM employees

WHERE employee\_id = p\_id;

RETURN v\_sal;

EXCEPTION

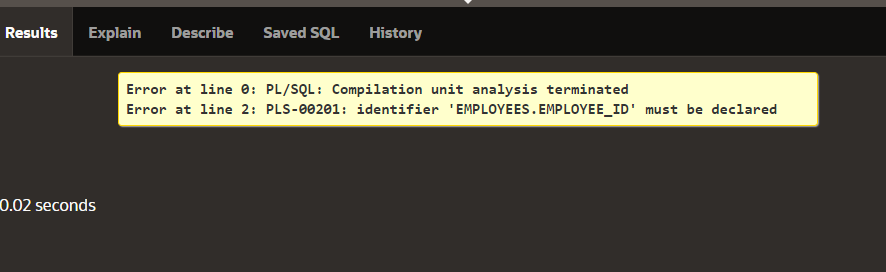
WHEN NO\_DATA\_FOUND THEN

RETURN NULL; -- or you could raise a custom exception or handle it in another way

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20001, 'More than one row found for the given employee\_id');

END get\_sal;



CREATE OR REPLACE FUNCTION get\_sal

(p\_id IN employees.employee\_id%TYPE) RETURN NUMBER IS

v\_sal employees.salary%TYPE := 0;

BEGIN

SELECT salary INTO v\_sal

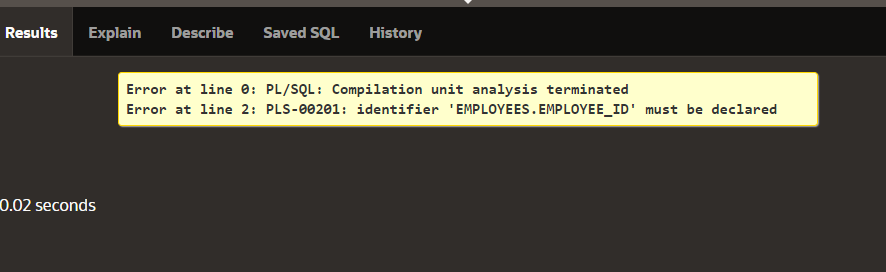
FROM employees WHERE employee\_id = p\_id;

RETURN v\_sal;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RETURN NULL;

END get\_sal;



DECLARE v\_sal employees.salary%type;

BEGIN

v\_sal := get\_sal(100); ...

END;

CREATE OR REPLACE FUNCTION valid\_dept

(p\_dept\_no departments.department\_id%TYPE)

RETURN BOOLEAN IS v\_valid VARCHAR2(1);

BEGIN

SELECT 'x‘ INTO v\_valid

FROM departments

WHERE department\_id = p\_dept\_no;

RETURN(true);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RETURN(false);

WHEN OTHERS THEN NULL;

END;

BEGIN …

IF valid\_dept(v\_departmentid) THEN

-- this was a valid department, so we’ll do this

part of the code, e.g. an insert into employees

ELSE

-- valid\_dept returned a false, so we are not

doing the insert

END IF;

…

END;

DECLARE v\_today DATE;

BEGIN

v\_today := SYSDATE; ...

END;

CREATE [OR REPLACE] PROCEDURE name [parameters] IS|AS

(Mandatory)

Variables, cursors, etc. (Optional)

BEGIN (Mandatory)

SQL and PL/SQL statements;

EXCEPTION (Optional)

WHEN exception-handling actions;

END [name]; (Mandatory)

CREATE [OR REPLACE] FUNCTION name [parameters] (Mandatory)

RETURN datatype IS|AS (Mandatory)

Variables, cursors, etc. (Optional)

BEGIN (Mandatory)

SQL and PL/SQL statements;

RETURN ...; (One Mandatory, more optional)

EXCEPTION (Optional)

WHEN exception-handling actions;

END [name]; (Mandatory)

CREATE OR REPLACE FUNCTION tax(p\_value IN NUMBER)

RETURN NUMBER IS

BEGIN

RETURN (p\_value \* 0.08);

END tax;

SELECT employee\_id, last\_name, salary, tax(salary)

FROM employees

WHERE department\_id = 50;

SELECT employee\_id, tax(salary)

FROM employees

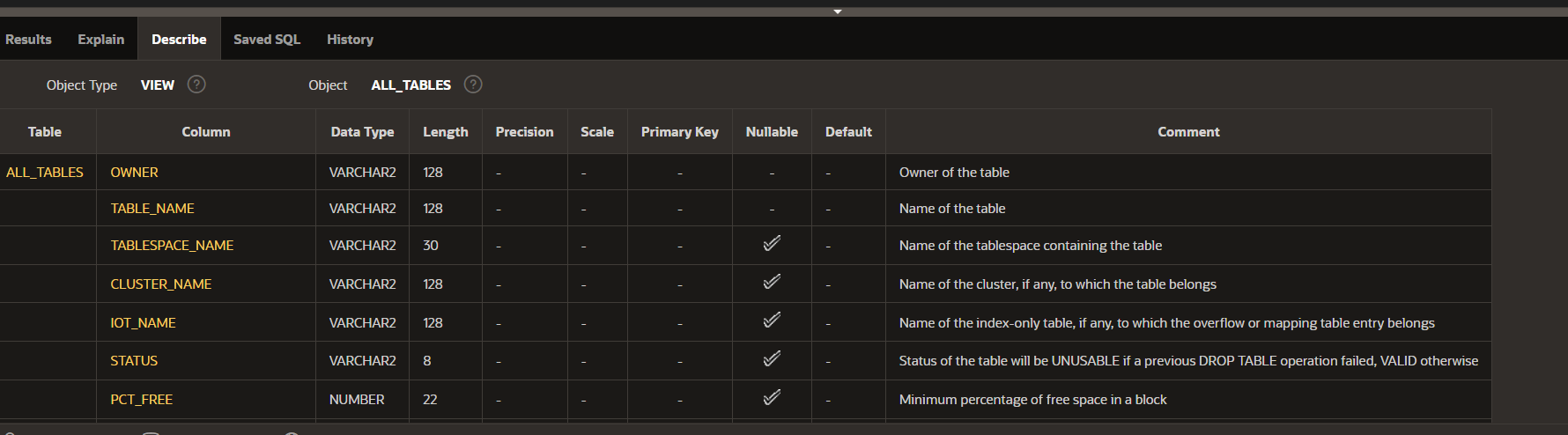
WHERE tax(salary) > (SELECT MAX(tax(salary))

FROM employees

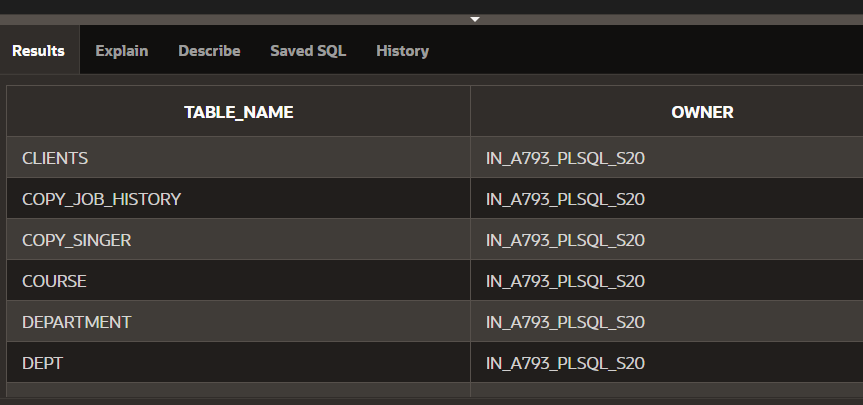
WHERE department\_id = 20)

ORDER BY tax(salary) DESC;

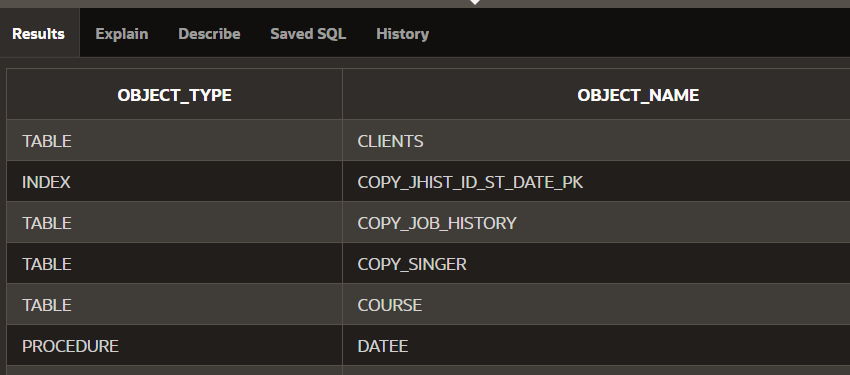
DESCRIBE ALL\_TABLES



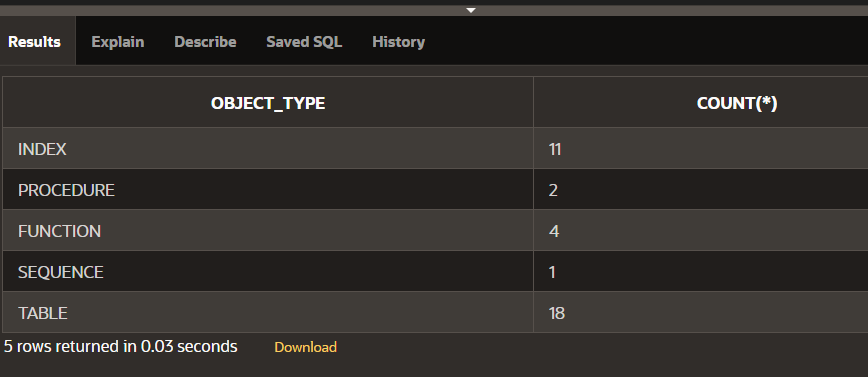
SELECT table\_name, owner FROM ALL\_TABLES;



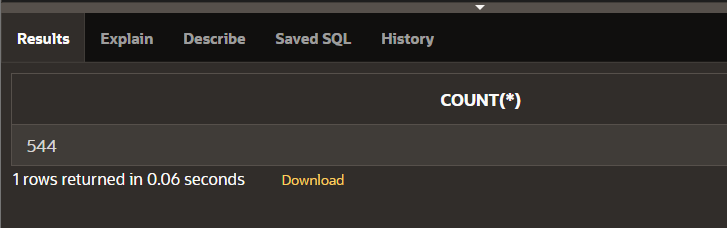
SELECT object\_type, object\_name FROM USER\_OBJECTS;



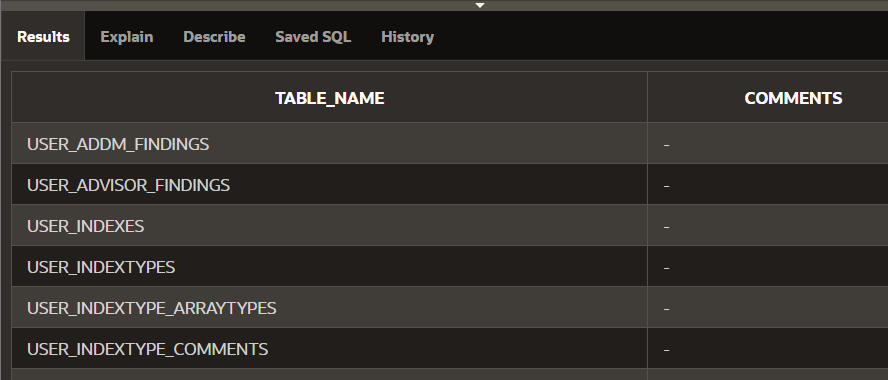
SELECT object\_type, COUNT(\*) FROM USER\_OBJECTS GROUP BY object\_type;



SELECT COUNT(\*) FROM DICT WHERE table\_name LIKE 'USER%';



SELECT \* FROM DICT WHERE table\_name LIKE 'USER%IND%';



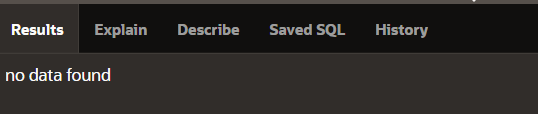
DROP {PROCEDURE procedure\_name | FUNCTION function\_name}

SELECT text

FROM USER\_SOURCE

WHERE name = 'TAX'

ORDER BY line;



CREATE OR REPLACE PACKAGE check\_emp\_pkg

IS

g\_max\_length\_of\_service CONSTANT NUMBER := 100;

PROCEDURE chk\_hiredate

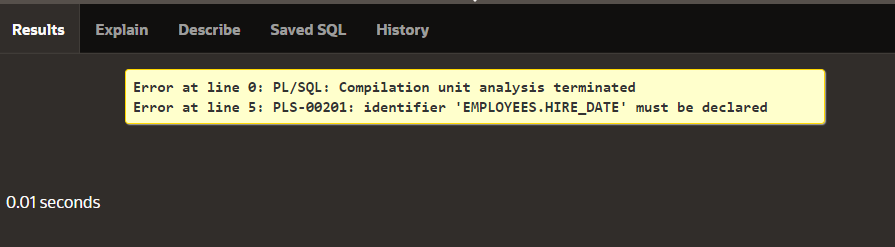
(p\_date IN employees.hire\_date%TYPE);

PROCEDURE chk\_dept\_mgr

(p\_empid IN employees.employee\_id%TYPE,

p\_mgr IN employees.manager\_id%TYPE);

END check\_emp\_pkg;



**SECTION 10**

CREATE OR REPLACE PACKAGE manage\_jobs\_pkg

IS

g\_todays\_date DATE := SYSDATE;

CURSOR jobs\_curs IS

SELECT employee\_id, job\_id FROM employees

ORDER BY employee\_id;

PROCEDURE update\_job

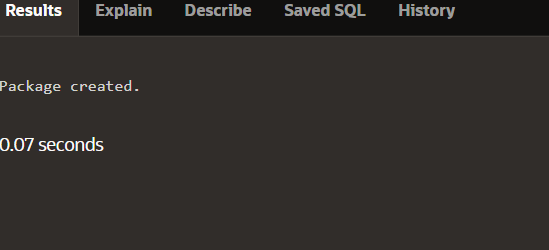
(p\_emp\_id IN employees.employee\_id%TYPE);

PROCEDURE fetch\_emps

(p\_job\_id IN employees.job\_id%TYPE,

p\_emp\_id OUT employees.employee\_id%TYPE);

END manage\_jobs\_pkg;



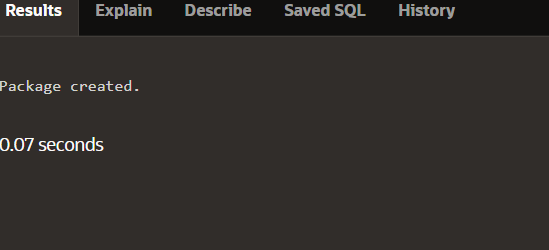
CREATE [OR REPLACE] PACKAGE BODY package\_name IS|AS

private type and variable declarations

subprogram bodies

[BEGIN initialization statements]

END [package\_name];



CREATE OR REPLACE PACKAGE BODY check\_emp\_pkg IS

PROCEDURE chk\_hiredate

(p\_date IN employees.hire\_date%TYPE)

IS BEGIN

IF MONTHS\_BETWEEN(SYSDATE, p\_date) >

g\_max\_length\_of\_service \* 12 THEN

RAISE\_APPLICATION\_ERROR(-20200, 'Invalid Hiredate');

END IF;

END chk\_hiredate;

PROCEDURE chk\_dept\_mgr

(p\_empid IN employees.employee\_id%TYPE,

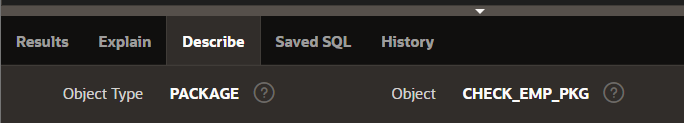
p\_mgr IN employees.manager\_id%TYPE)

IS BEGIN ...

END chk\_dept\_mgr;

END check\_emp\_pkg;

DESCRIBE check\_emp\_pkg



CREATE OR REPLACE PACKAGE salary\_pkg

IS

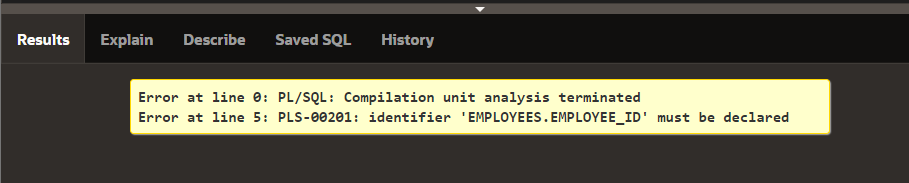
g\_max\_sal\_raise CONSTANT NUMBER := 0.20;

PROCEDURE update\_sal

(p\_employee\_id employees.employee\_id%TYPE,

p\_new\_salary employees.salary%TYPE);

END salary\_pkg;



PROCEDURE update\_sal -- public procedure

(p\_employee\_id employees.employee\_id%TYPE,

p\_new\_salary employees.salary%TYPE)

IS v\_old\_salary employees.salary%TYPE; -- local variable

BEGIN

SELECT salary INTO v\_old\_salary FROM employees

WHERE employee\_id = p\_employee\_id;

IF validate\_raise(v\_old\_salary, p\_new\_salary) THEN

UPDATE employees

SET salary = p\_new\_salary

WHERE employee\_id = p\_employee\_id;

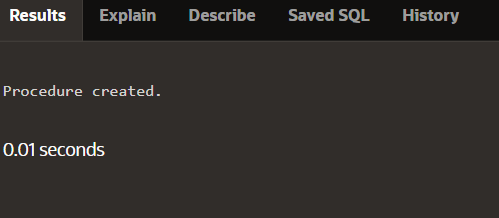
ELSE

RAISE\_APPLICATION\_ERROR(-20210, 'Raise too high');

END IF;

END update\_sal;

END salary\_pkg;



DECLARE

v\_bool BOOLEAN;

v\_number NUMBER;

BEGIN

salary\_pkg.update\_sal(100,25000); -- 1

update\_sal(100,25000); -- 2

v\_bool := salary\_pkg.validate\_raise(24000,25000); -- 3

v\_number := salary\_pkg.g\_max\_sal\_raise; -- 4

v\_number := salary\_pkg.v\_old\_salary; -- 5

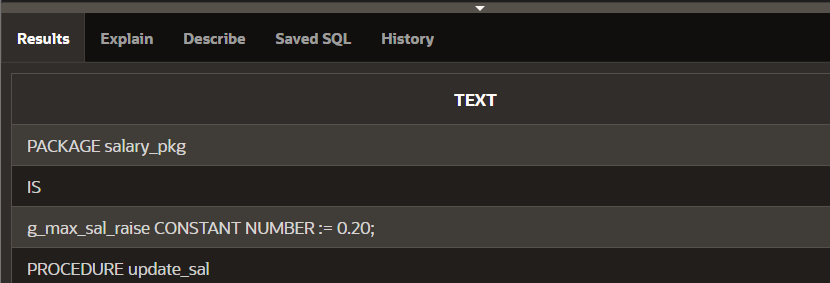
END;

SELECT text

FROM user\_source

WHERE name = 'SALARY\_PKG' AND type = 'PACKAGE'

ORDER BY line;

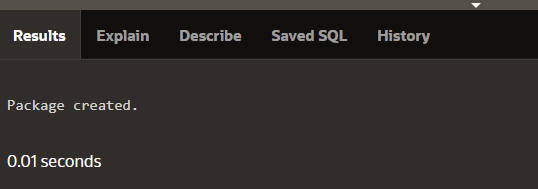


CREATE PACKAGE sample\_pack IS

PROCEDURE sample\_proc (p\_char\_param IN CHAR);

PROCEDURE sample\_proc (p\_varchar\_param IN VARCHAR2);

END sample\_pack;



CREATE OR REPLACE PACKAGE dept\_pkg IS

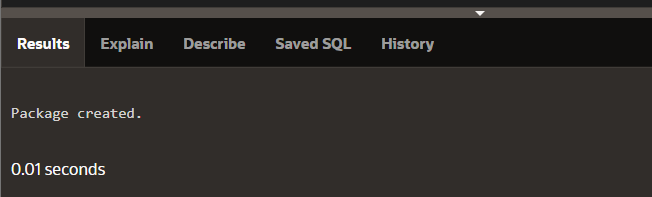
PROCEDURE add\_department(p\_deptno NUMBER,

p\_name VARCHAR2 := 'unknown', p\_loc NUMBER := 1700);

PROCEDURE add\_department(

p\_name VARCHAR2 := 'unknown', p\_loc NUMBER := 1700);

END dept\_pkg;



CREATE OR REPLACE PACKAGE BODY dept\_pkg IS

PROCEDURE add\_department (p\_deptno NUMBER,

p\_name VARCHAR2:='unknown', p\_loc NUMBER:=1700) IS

BEGIN

INSERT INTO departments(department\_id,

department\_name, location\_id)

VALUES (p\_deptno, p\_name, p\_loc);

END add\_department;

PROCEDURE add\_department (

p\_name VARCHAR2:='unknown', p\_loc NUMBER:=1700) IS

BEGIN

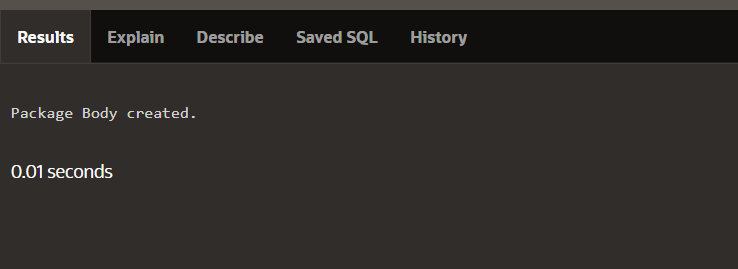
INSERT INTO departments (department\_id,

department\_name, location\_id)

VALUES (departments\_seq.NEXTVAL, p\_name, p\_loc);

END add\_department;

END dept\_pkg;

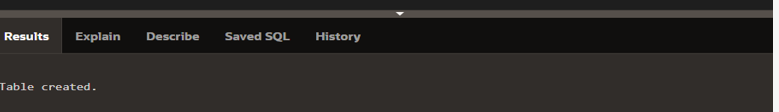


**SECTION 11**

CREATE TABLE employees (

employee\_id NUMBER

);



BEGIN

INSERT INTO employees (employee\_id) VALUES (1);

INSERT INTO employees (employee\_id) VALUES (2);

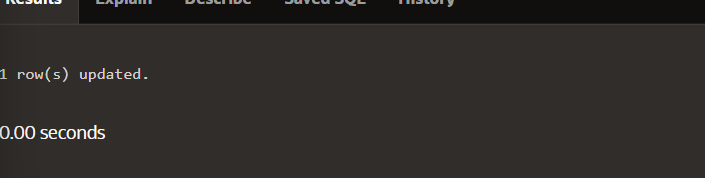
INSERT INTO employees (employee\_id) VALUES (3);

INSERT INTO employees (employee\_id) VALUES (4);

INSERT INTO employees (employee\_id) VALUES (5);

COMMIT;

END;



CREATE OR REPLACE PACKAGE curs\_pkg IS

CURSOR emp\_curs IS SELECT employee\_id FROM employees

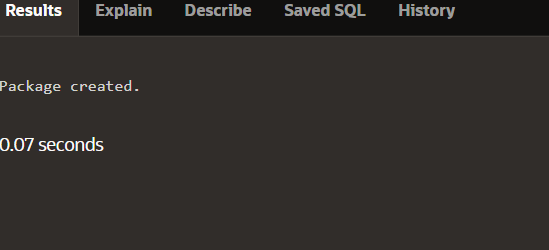
ORDER BY employee\_id;

PROCEDURE open\_curs;

FUNCTION fetch\_n\_rows(n NUMBER := 1) RETURN BOOLEAN;

PROCEDURE close\_curs;

END curs\_pkg;



CREATE OR REPLACE PACKAGE BODY curs\_pkg IS

PROCEDURE open\_curs IS

BEGIN

IF NOT emp\_curs%ISOPEN THEN OPEN emp\_curs; END IF;

END open\_curs;

FUNCTION fetch\_n\_rows(n NUMBER := 1) RETURN BOOLEAN IS

emp\_id employees.employee\_id%TYPE;

BEGIN

FOR count IN 1 .. n LOOP

FETCH emp\_curs INTO emp\_id;

EXIT WHEN emp\_curs%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Id: ' ||(emp\_id));

END LOOP;

RETURN emp\_curs%FOUND;

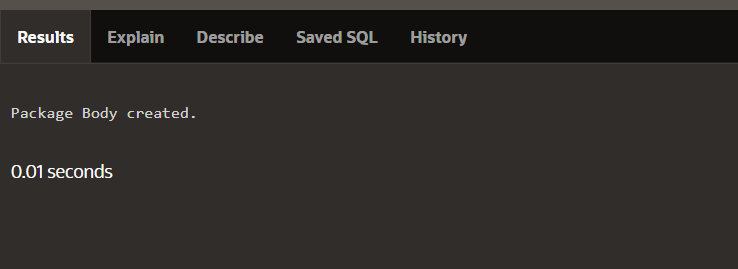
END fetch\_n\_rows;

PROCEDURE close\_curs IS BEGIN

IF emp\_curs%ISOPEN THEN CLOSE emp\_curs; END IF;

END close\_curs;

END curs\_pkg;



DECLARE

v\_more\_rows\_exist BOOLEAN := TRUE;

BEGIN

curs\_pkg.open\_curs; --1

LOOP

v\_more\_rows\_exist := curs\_pkg.fetch\_n\_rows(3); --2

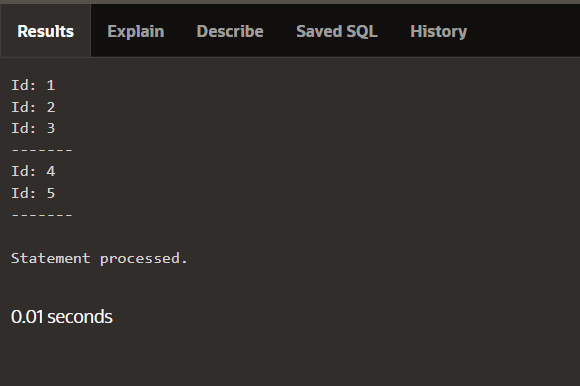
DBMS\_OUTPUT.PUT\_LINE('-------');

EXIT WHEN NOT v\_more\_rows\_exist;

END LOOP;

curs\_pkg.close\_curs; --3

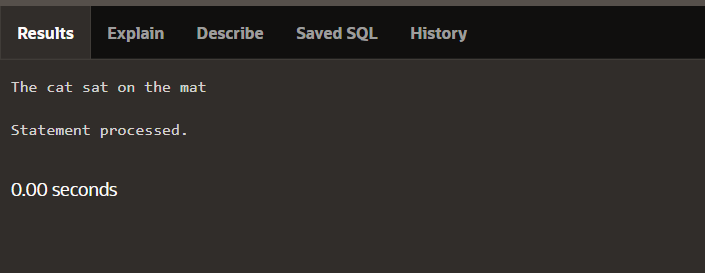
END;



BEGIN

DBMS\_OUTPUT.PUT\_LINE('The cat sat on the mat');

END;



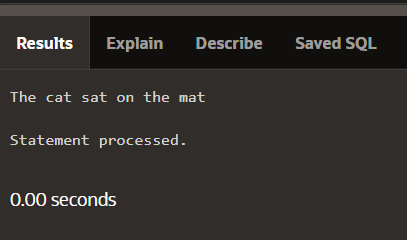
BEGIN

DBMS\_OUTPUT.PUT('The cat sat ');

DBMS\_OUTPUT.PUT('on the mat');

DBMS\_OUTPUT.NEW\_LINE;

END;



DECLARE

v\_bool1 BOOLEAN := true;

v\_bool2 BOOLEAN := false;

v\_number NUMBER;

BEGIN

IF v\_bool1 AND NOT v\_bool2 AND v\_number < 25 THEN

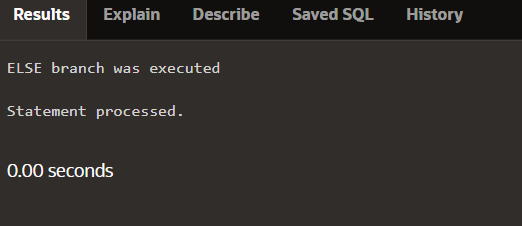
DBMS\_OUTPUT.PUT\_LINE('IF branch was executed');

ELSE

DBMS\_OUTPUT.PUT\_LINE('ELSE branch was executed');

END IF;

END;

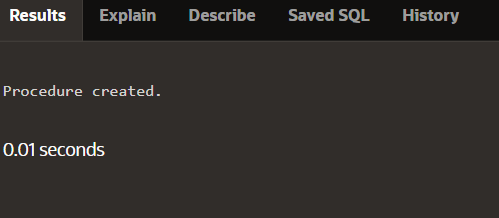


CREATE OR REPLACE PROCEDURE do\_some\_work IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('string');

END;



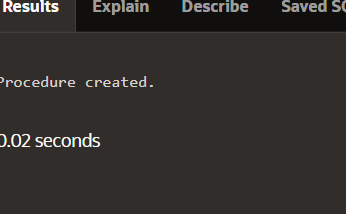
CREATE OR REPLACE PROCEDURE do\_some\_work

( p\_output OUT VARCHAR2)

IS BEGIN

p\_output := 'string';

END;



CREATE SYNONYM UTL\_FILE FOR SYS.UTL\_FILE;

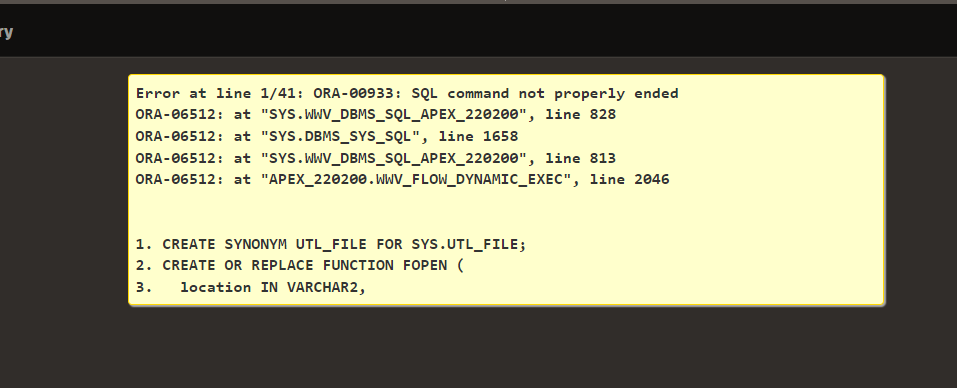
CREATE OR REPLACE FUNCTION FOPEN (

location IN VARCHAR2,

filename IN VARCHAR2,

open\_mode IN VARCHAR2

) RETURN UTL\_FILE.FILE\_TYPE;



PROCEDURE read(dir VARCHAR2, filename VARCHAR2) IS

file UTL\_FILE.FILE\_TYPE;

BEGIN

IF NOT UTL\_FILE.IS\_OPEN(file) THEN

file := UTL\_FILE.FOPEN (dir, filename, 'r');

END IF; ...

END read;



CREATE OR REPLACE PROCEDURE send\_mail\_with\_text

(p\_text\_attachment IN VARCHAR2)

IS BEGIN

UTL\_MAIL.SEND\_ATTACH\_VARCHAR2(

sender => 'me@here.com',

recipients => 'you@somewhere.net',

message => 'See attachment',

subject => 'Useful document for our project',

attachment => p\_text\_attachment,

END send\_mail\_with\_text;

**SECTION 12**

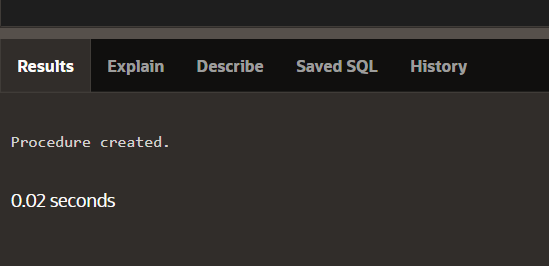
CREATE OR REPLACE PROCEDURE drop\_any\_table(p\_table\_name IN VARCHAR2)

IS

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE ' || p\_table\_name;

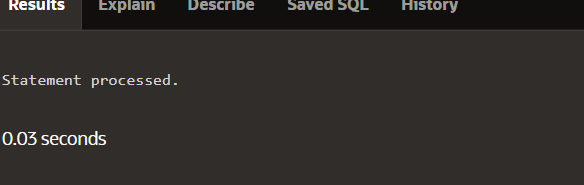
END;



BEGIN

drop\_any\_table('EMPLOYEES');

END;



CREATE FUNCTION del\_rows(p\_table\_name VARCHAR2)

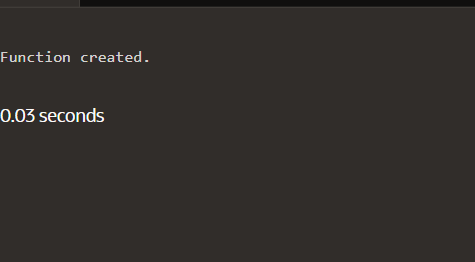
RETURN NUMBER IS

BEGIN

EXECUTE IMMEDIATE 'DELETE FROM ' || p\_table\_name;

RETURN SQL%ROWCOUNT;

END;



DECLARE

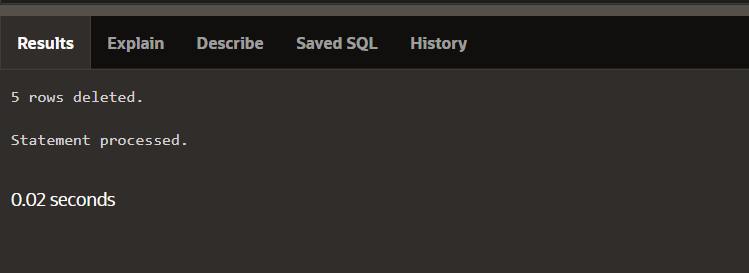
v\_count NUMBER;

BEGIN

v\_count := del\_rows('employee');

DBMS\_OUTPUT.PUT\_LINE(v\_count || ' rows deleted.');

END;



CREATE PROCEDURE add\_row(p\_table\_name VARCHAR2,

p\_id NUMBER, p\_name VARCHAR2) IS

BEGIN

EXECUTE IMMEDIATE 'INSERT INTO ' || p\_table\_name ||

'VALUES(' || p\_id || ', ''' || p\_name || ''')';

END;



CREATE PROCEDURE compile\_plsql

(p\_name VARCHAR2,p\_type VARCHAR2,p\_options VARCHAR2 := NULL)

IS

v\_stmt VARCHAR2(200);

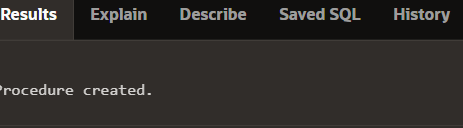
BEGIN

v\_stmt := 'ALTER ' || p\_type || ' ' || p\_name || ' COMPILE'

|| ' ' || p\_options;

EXECUTE IMMEDIATE v\_stmt;

END;



CREATE OR REPLACE FUNCTION del\_rows

(p\_table\_name VARCHAR2) RETURN NUMBER IS

v\_csr\_id INTEGER;

v\_rows\_del NUMBER;

BEGIN

v\_csr\_id := DBMS\_SQL.OPEN\_CURSOR;

DBMS\_SQL.PARSE(v\_csr\_id,

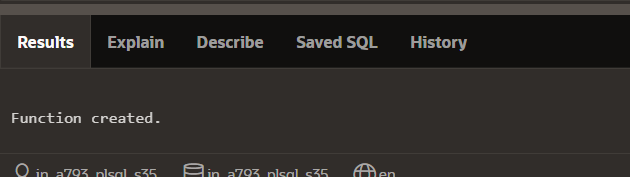
'DELETE FROM ' || p\_table\_name, DBMS\_SQL.NATIVE);

v\_rows\_del := DBMS\_SQL.EXECUTE(v\_csr\_id);

DBMS\_SQL.CLOSE\_CURSOR(v\_csr\_id);

RETURN v\_rows\_del;

END;



CREATE OR REPLACE PACKAGE emp\_pkg IS

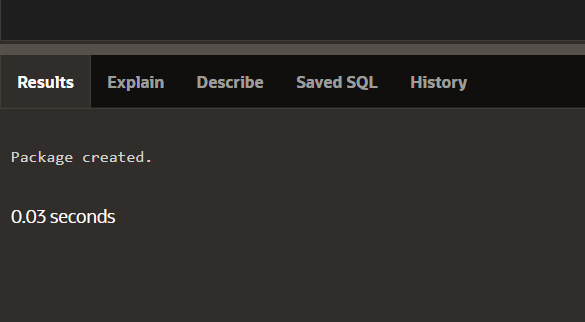
TYPE t\_emp IS TABLE OF employee%ROWTYPE

INDEX BY BINARY\_INTEGER;

PROCEDURE emp\_proc

(p\_small\_arg IN NUMBER, p\_big\_arg OUT t\_emp);

END emp\_pkg;



CREATE OR REPLACE PACKAGE emp\_pkg IS

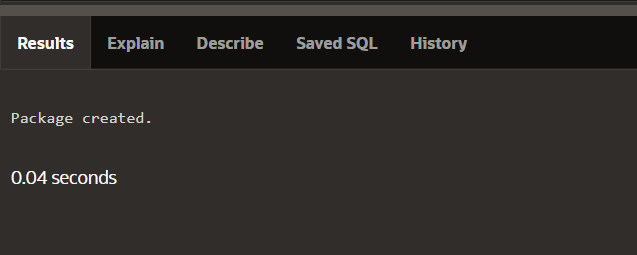
TYPE t\_emp IS TABLE OF employee%ROWTYPE

INDEX BY BINARY\_INTEGER;

PROCEDURE emp\_proc

(p\_small\_arg IN NUMBER, p\_big\_arg OUT t\_emp);

END emp\_pkg;



CREATE OR REPLACE PACKAGE emp\_pkg IS

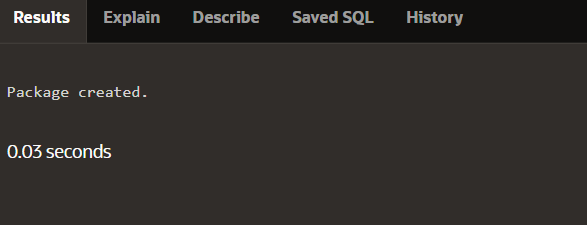
TYPE t\_emp IS TABLE OF employee%ROWTYPE

INDEX BY BINARY\_INTEGER;

PROCEDURE emp\_proc

(p\_small\_arg IN NUMBER, p\_big\_arg OUT NOCOPY t\_emp);

END emp\_pkg;



CREATE OR REPLACE PACKAGE emp\_pkg IS

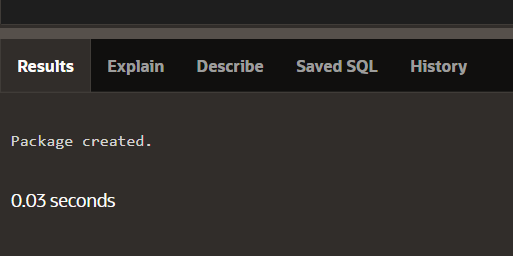
TYPE t\_emp IS TABLE OF employee%ROWTYPE

INDEX BY BINARY\_INTEGER;

PROCEDURE emp\_proc

(p\_small\_arg IN NUMBER, p\_big\_arg OUT NOCOPY t\_emp);

END emp\_pkg;



CREATE OR REPLACE FUNCTION twicenum

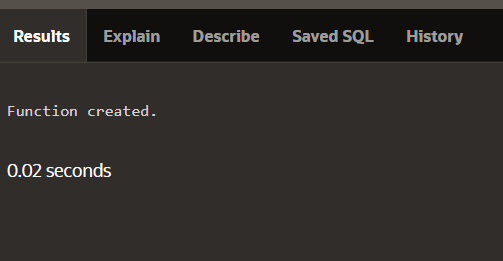
(p\_number IN NUMBER)

RETURN NUMBER IS

BEGIN

RETURN p\_number \* 2;

END twicenum;



CREATE OR REPLACE PROCEDURE fetch\_all\_emps IS

TYPE t\_emp IS TABLE OF employees%ROWTYPE INDEX BY BINARY\_INTEGER;

v\_emptab t\_emp;

BEGIN

SELECT \* BULK COLLECT INTO v\_emptab FROM employees;

FOR i IN v\_emptab.FIRST..v\_emptab.LAST LOOP

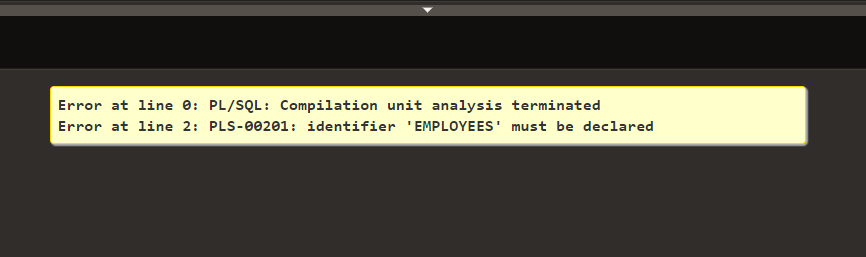
IF v\_emptab.EXISTS(i) THEN

DBMS\_OUTPUT.PUT\_LINE(v\_emptab(i).last\_name);

END IF;

END LOOP;

END fetch\_all\_emps;



CREATE OR REPLACE PROCEDURE update\_all\_emps

(p\_salary\_raise\_percent IN NUMBER) IS

TYPE t\_empid IS TABLE OF employees.employee\_id%TYPE

INDEX BY BINARY\_INTEGER;

TYPE t\_sal IS TABLE OF employees.salary%TYPE

INDEX BY BINARY\_INTEGER;

v\_empidtab t\_empid;

v\_saltab t\_sal;

BEGIN

SELECT employee\_id BULK COLLECT INTO v\_empidtab FROM employees;

FORALL i IN v\_empidtab.FIRST..v\_empidtab.LAST

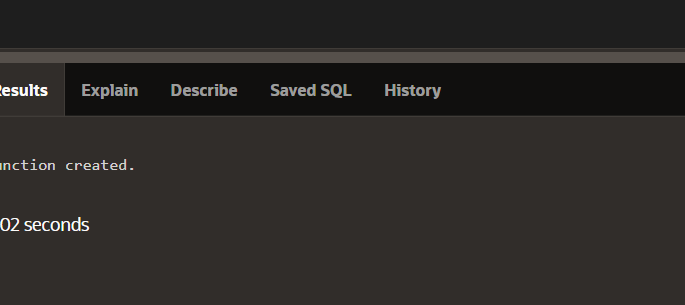
UPDATE employees

SET salary = salary \* (1 + p\_salary\_raise\_percent)

WHERE employee\_id = v\_empidtab(i)

RETURNING salary BULK COLLECT INTO v\_saltab;

END update\_all\_emps;



CREATE OR REPLACE PROCEDURE update\_one\_emp

(p\_emp\_id IN employees.employee\_id%TYPE,

p\_salary\_raise\_percent IN NUMBER) IS

v\_new\_salary employees.salary%TYPE;

BEGIN

UPDATE employees

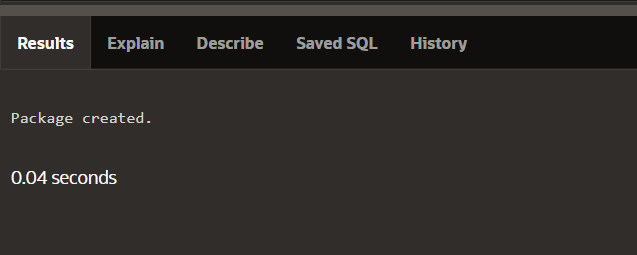
SET salary = salary \* (1 + p\_salary\_raise\_percent)

WHERE employee\_id = p\_emp\_id

RETURNING salary INTO v\_new\_salary;

DBMS\_OUTPUT.PUT\_LINE('New salary is: ' || v\_new\_salary);

END update\_one\_emp;



CREATE OR REPLACE PROCEDURE update\_one\_emp

(p\_emp\_id IN employees.employee\_id%TYPE,

p\_salary\_raise\_percent IN NUMBER) IS

v\_new\_salary employees.salary%TYPE;

BEGIN

UPDATE employees

SET salary = salary \* (1 + p\_salary\_raise\_percent)

WHERE employee\_id = p\_emp\_id;

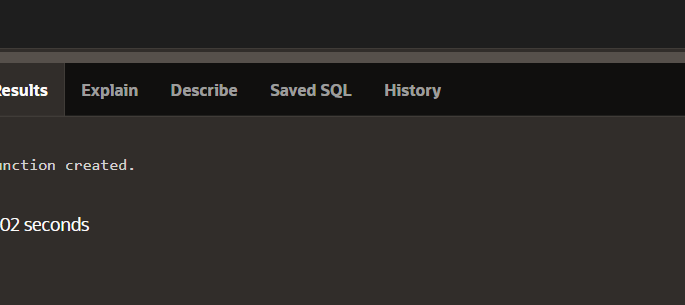
SELECT salary INTO v\_new\_salary

FROM employees

WHERE employee\_id = p\_emp\_id;

DBMS\_OUTPUT.PUT\_LINE('New salary is: ' || v\_new\_salary);

END update\_one\_emp;



CREATE OR REPLACE PROCEDURE insert\_emps IS

TYPE t\_emps IS TABLE OF employees%ROWTYPE INDEX BY BINARY\_INTEGER;

v\_emptab t\_emps;

BEGIN

SELECT \* BULK COLLECT INTO v\_emptab FROM employees;

FORALL i IN v\_emptab.FIRST..v\_emptab.LAST SAVE EXCEPTIONS

INSERT INTO employees VALUES v\_emptab(i);

EXCEPTION

WHEN OTHERS THEN

FOR j in 1..SQL%BULK\_EXCEPTIONS.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(SQL%BULK\_EXCEPTIONS(j).ERROR\_INDEX);

DBMS\_OUTPUT.PUT\_LINE(SQL%BULK\_EXCEPTIONS(j).ERROR\_CODE);

END LOOP;

END insert\_emps;

**SECTION 14**

SELECT name, type, referenced\_name, referenced\_type

FROM user\_dependencies

WHERE referenced\_name IN ('EMPLOYEES','EMP\_VW' );

