Fusneica Florentin-Cristian, grupa 241

PLSQL1 EX1

```
SET SERVEROUTPUT ON;
SET VERIFY OFF;

DECLARE
numar number(3):=100;
mesaj1 varchar2(255):='text 1';
mesaj2 varchar2(255):='text 2';
BEGIN
 DECLARE
 numar number(3):=1;
 mesaj1 varchar2(255):='text 2';
 mesaj2 varchar2(255):='text 3';
 BEGIN
 numar:=numar+1;
 --a--
 DBMS_OUTPUT.PUT_LINE(numar);
 mesaj2:=mesaj2||' adaugat in sub-bloc';
 END;
numar:=numar+1;
mesaj1:=mesaj1||' adaugat un blocul principal';
--b--
 DBMS_OUTPUT.PUT_LINE(mesaj1);
mesaj2:=mesaj2||' adaugat in blocul principal';
--c--
DBMS_OUTPUT.PUT_LINE(mesaj2);
 numar := numar + 1;
 -- d) --
DBMS_OUTPUT.PUT_LINE(numar);
mesaj1 := mesaj1 || ' adaugat in blocul principal';
-- e)--
 DBMS_OUTPUT.PUT_LINE(mesaj1);
 mesaj2 := mesaj2 || ' adaugat in blocul principal';
 -- f) --
 DBMS_OUTPUT.PUT_LINE(mesaj2);
END;
/
```

0.43799999 seconds

Worksheet | Query Builder

```
 numar:=numar+1;
 --a--
 DBMS_OUTPUT.PUT_LINE(numar);
 mesaj2:=mesaj2||' adaugat in sub-bloc';
 END;
numar:=numar+1;
mesaj1:=mesaj1||' adaugat un blocul principal';
--b--
 DBMS_OUTPUT.PUT_LINE(mesaj1);
mesaj2:=mesaj2||' adaugat in blocul principal';
--c--
DBMS_OUTPUT.PUT_LINE(mesaj2);
 numar := numar + 1;
 -- d) --
DBMS_OUTPUT.PUT_LINE(numar);
mesaj1 := mesaj1 || ' adaugat in blocul principal';
-- e)--
 DBMS_OUTPUT.PUT_LINE(mesaj1);
 mesaj2 := mesaj2 || ' adaugat in blocul principal';
 -- f) --
 DBMS_OUTPUT.PUT_LINE(mesaj2);
END;
/
```

Script Output

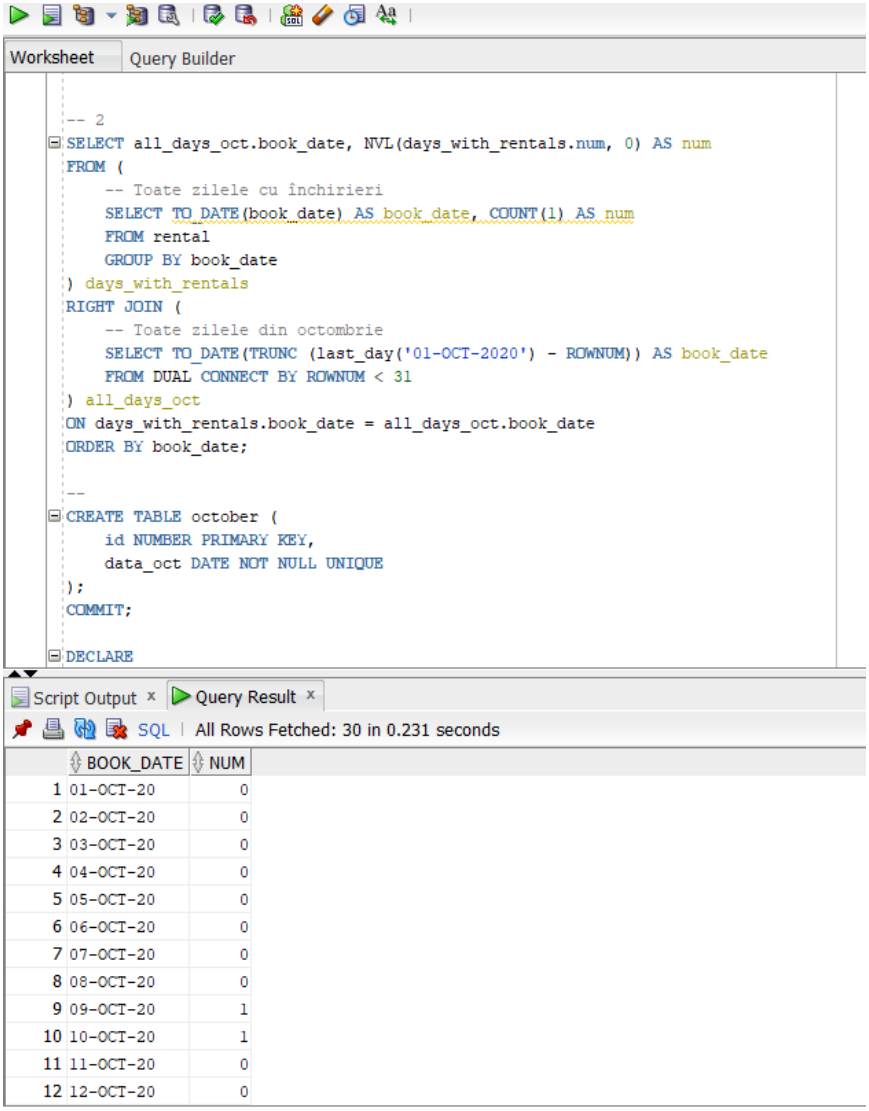
Task completed in 0.438 seconds

```
text 1 adaugat un blocul principal
text 2 adaugat in blocul principal
102
text 1 adaugat un blocul principal adaugat in blocul principal
text 2 adaugat in blocul principal adaugat in blocul principal


PL/SQL procedure successfully completed.
```

PLSQL1 EX2

```
-- 2
--a
SELECT all_days_oct.book_date, NVL(days_with_rentals.num, 0) AS num
FROM (
   -- Toate zilele cu închirieri
   SELECT TO_DATE(book_date) AS book_date, COUNT(1) AS num
   FROM rental
   GROUP BY book_date
) days_with_rentals
RIGHT JOIN (
   -- Toate zilele din octombrie
   SELECT TO_DATE(TRUNC (last_day('01-OCT-2020') - ROWNUM)) AS book_date
   FROM DUAL CONNECT BY ROWNUM < 31
) all_days_oct
ON days_with_rentals.book_date = all_days_oct.book_date
```

ORDER BY book_date;



--b- -

--

Crearea tabelei October

```
CREATE TABLE october (
   id NUMBER PRIMARY KEY,
   data_oct DATE NOT NULL UNIQUE
);
COMMIT;

DECLARE
   day_o DATE := '01-OCT-2020';
BEGIN
--Parcurgerea zilelor din luna octombrie
   FOR i IN 1..31 LOOP
      INSERT INTO october VALUES (i, day_o);
      day_o := day_o + 1;
   END LOOP;
END;
/
```

```
SELECT data_oct, NVL(num, 0)
FROM october
LEFT JOIN (
--selectarea datelor cu inchirieri
   SELECT TO_DATE(book_date) AS data_oct, COUNT(1) AS num
   FROM rental
   GROUP BY book_date
)
USING (data_oct)
ORDER BY data_oct;
```

```
--b--
--Crearea tabelei October
CREATE TABLE october (
    id NUMBER PRIMARY KEY,
    data_oct DATE NOT NULL UNIQUE
);
COMMIT;

DECLARE
    day_o DATE := '01-OCT-2020';
BEGIN
--Parcurgerea zilelor din luna octombrie
    FOR i IN 1..31 LOOP
        INSERT INTO october VALUES (i, day_o);
        day_o := day_o + 1;
    END LOOP;
END;
/
```

Script Output × | Query Result ×

Task completed in 0.335 seconds

```
Table OCTOBER created.


Commit complete.


PL/SQL procedure successfully completed.
```

```sql
/
SELECT data_oct, NVL(num, 0)
FROM october
LEFT JOIN (
    SELECT TO_DATE(book_date) AS data_oct, COUNT(1) AS num
    FROM rental
    GROUP BY book_date
)
USING (data_oct)
ORDER BY data_oct;
```

Script Output × | Query Result ×

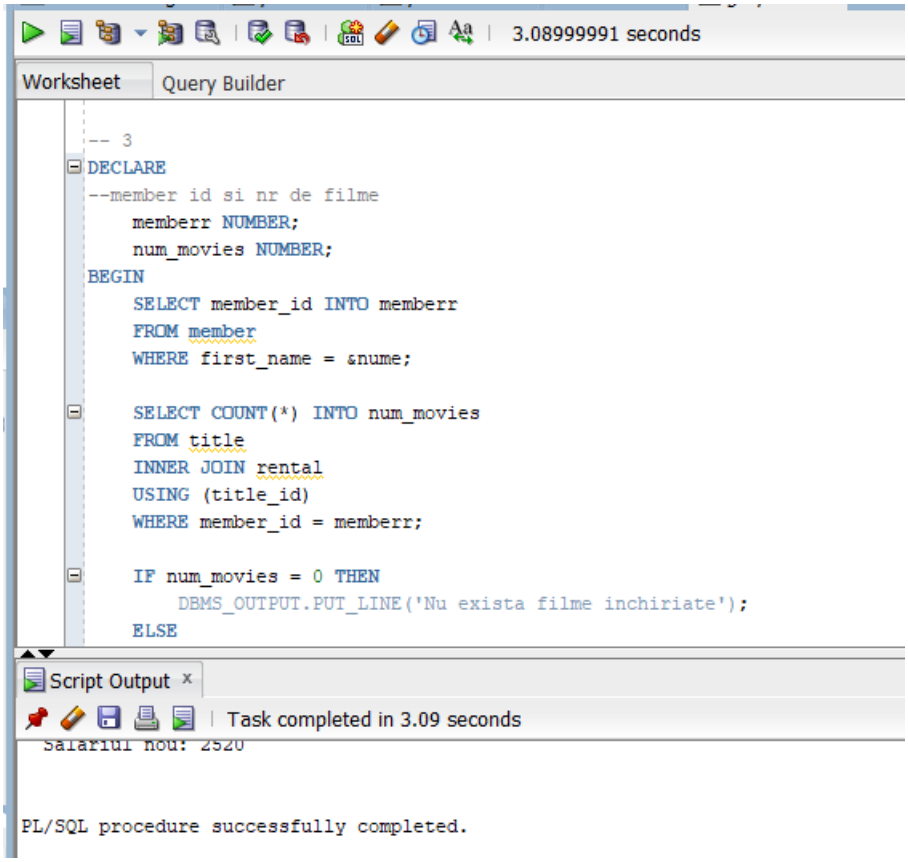
SQL | All Rows Fetched: 31 in 0.034 seconds

	DATA_OCT	NVL(NUM,0)
7	07-OCT-20	0
8	08-OCT-20	0
9	09-OCT-20	1
10	10-OCT-20	1
11	11-OCT-20	0
12	12-OCT-20	0
13	13-OCT-20	0

PLSQL1 EX 3

```sql
-- 3
DECLARE
--member id si nr de filme
   memberr NUMBER;
   num_movies NUMBER;
BEGIN
   SELECT member_id INTO memberr
   FROM member
   WHERE first_name = &nume;

   SELECT COUNT(*) INTO num_movies
   FROM title
   INNER JOIN rental
   USING (title_id)
   WHERE member_id = memberr;

   IF num_movies = 0 THEN
      DBMS_OUTPUT.PUT_LINE('Nu exista filme inchiriate');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Filme inchiriate: ' || num_movies);
   END IF;

EXCEPTION
   WHEN too_many_rows THEN
      DBMS_OUTPUT.PUT_LINE('Mai multi oameni cu acelasi nume');

   when no_data_found then
    DBMS_OUTPUT.PUT_LINE('Nu exista membrul cautat');
```

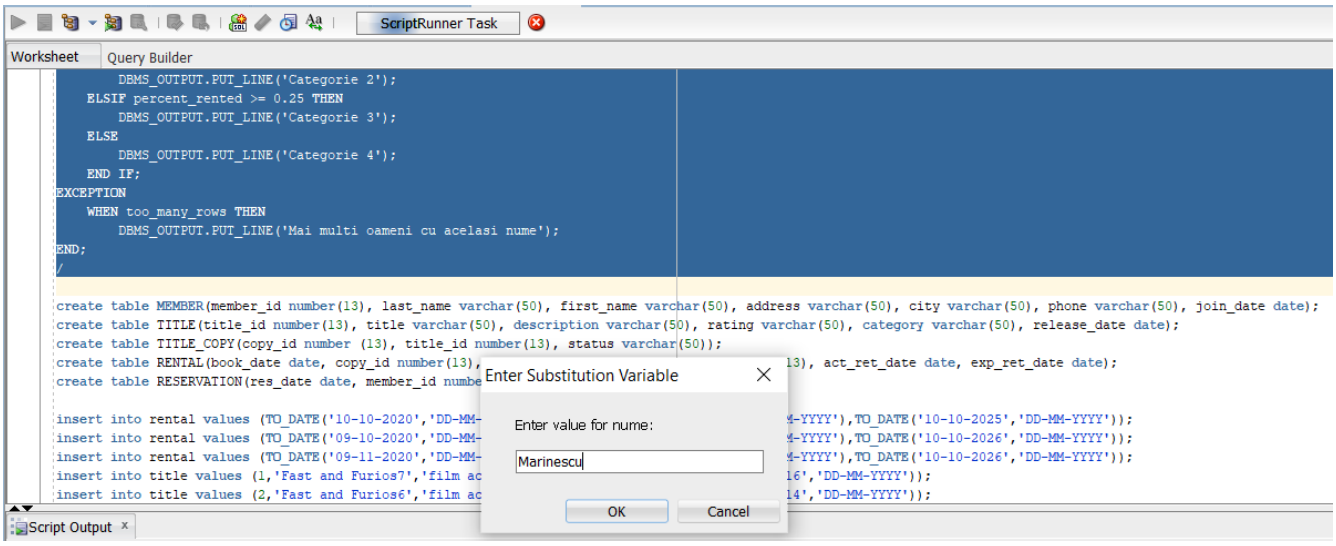
END;
/



PLSQL1 EX 4

```
-- 4
DECLARE
   memberr NUMBER;
   num_movies NUMBER;
   total_movies NUMBER;
   percent_rented NUMBER;
BEGIN
   SELECT member_id INTO memberr
   FROM member
   WHERE first_name = &nume;

   SELECT COUNT(*) INTO num_movies
   FROM title
   INNER JOIN rental
   USING (title_id)
   WHERE member_id = memberr;

   IF num_movies = 0 THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Nu exista filme inchiriate');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Filme inchiriate: ' || num_movies);
    END IF;

    SELECT COUNT(*) INTO total_movies
    FROM title;

    percent_rented := num_movies / total_movies;
    IF percent_rented >= 0.75 THEN
        DBMS_OUTPUT.PUT_LINE('Categorie 1');
    ELSIF percent_rented >= 0.5 THEN
        DBMS_OUTPUT.PUT_LINE('Categorie 2');
    ELSIF percent_rented >= 0.25 THEN
        DBMS_OUTPUT.PUT_LINE('Categorie 3');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Categorie 4');
    END IF;
EXCEPTION
    WHEN too_many_rows THEN
        DBMS_OUTPUT.PUT_LINE('Mai multi oameni cu acelasi nume');
END;
/
```



PLSQL1 EX 5

-- 5

```
DROP TABLE membru_discount;

CREATE TABLE membru_discount AS (
   SELECT * FROM member
);


ALTER TABLE membru_discount
ADD discount NUMBER DEFAULT NULL;

DECLARE
   membru NUMBER;
   num_filme NUMBER;
   total NUMBER;
   procent NUMBER;
   discount_in NUMBER;
   discount_nou NUMBER;
BEGIN
--codul introdus de la tastatura
   membru := &cod_membru;
--selectarea numarului de filme
   SELECT COUNT(*) INTO num_filme
   FROM title
   INNER JOIN rental
   USING (title_id)
   WHERE member_id = membru;
--numarul total de filme existente
   SELECT COUNT(*) INTO total
   FROM title;
   SELECT discount INTO discount_in
   FROM membru_discount
   WHERE member_id = membru;
--calcularea procentului de inchiriere
   procent := num_filme / total;
   --calcularea procentului in functie de categorie
   IF procent >= 0.75 THEN
      discount_nou := 0.10;
   ELSIF procent >= 0.5 THEN
      discount_nou := 0.05;
   ELSIF procent >= 0.25 THEN
     discount_nou := 0.03;
   ELSE
      discount_nou := NULL;
   END IF;

   IF NVL(discount_in, 0) != NVL(discount_nou, 0) THEN
      UPDATE membru_discount
      SET discount = discount_nou
      WHERE member_id = membru;
```

```
        DBMS_OUTPUT.PUT_LINE('Noul discount ' || discount_nou);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Discount neschimbat');
    END IF;
END;
/
```

2.93300009 seconds

Worksheet | Query Builder

```
        DBMS_OUTPUT.PUT_LINE('Multiple people with same name');
END;
/

-- 5
DROP TABLE membru_discount;

CREATE TABLE membru_discount AS (
    SELECT * FROM member
);


ALTER TABLE membru_discount
ADD discount NUMBER DEFAULT NULL;

DECLARE
    membru NUMBER;
    num_filme NUMBER;
    total NUMBER;
    procent NUMBER;
    discount_in NUMBER;
    discount_nou NUMBER;
BEGIN
--codul introdus de la tastatura
    membru := &cod_membru;
--selectarea numarului de filme
    SELECT COUNT(*) INTO num_filme
    FROM title
    INNER JOIN rental
    USING (title_id)
```

Script Output | Query Result

Task completed in 2.933 seconds

```
Table MEMBRU_DISCOUNT created.


Table MEMBRU_DISCOUNT altered.


PL/SQL procedure successfully completed.
```

```
Discount neschimbat


PL/SQL procedure successfully completed.
```

```
select * from membru_discount where member_id=1;
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.023 seconds

	MEMBER_ID	LAST_NAME	FIRST_NAME	ADDRESS	CITY	PHONE	JOIN_DATE	DISCOUNT
1	1	Marinescu	Antonia	Str Lalelelor	Bucuresti	0744345656	04-OCT-18	0.1

PLSQL2 EX 1

```
-- 1

-- fac o copie a tabelei employees pt a putea face modificari pe ea
DROP TABLE empl_copie;
CREATE TABLE empl_copie AS (SELECT * FROM employees);
COMMIT;

DECLARE
   -- Colecție cu idurile celor care nu castiga comision
   TYPE emp_ids_vector IS VARRAY(5)
      OF empl_copie.employee_id%TYPE NOT NULL;
   emps emp_ids_vector := emp_ids_vector();
   emp_id empl_copie.employee_id%TYPE;
   emp_salary empl_copie.salary%TYPE;
BEGIN
   SELECT *
   BULK COLLECT INTO emps
   FROM
   (
      SELECT employee_id
      FROM empl_copie
      WHERE commission_pct IS NULL
      ORDER BY salary ASC
   )
   WHERE rownum <= 5;

   FOR i IN emps.FIRST..emps.LAST LOOP
      emp_id := emps(i);

      DBMS_OUTPUT.PUT_LINE('Actualizez salariatul cu id ul' || emp_id);

      SELECT salary
      INTO emp_salary
      FROM empl_copie
      WHERE employee_id = emp_id;

      DBMS_OUTPUT.PUT_LINE('  Salariu vechi: ' || emp_salary);
--maresc salariul cu 5%
      UPDATE empl_copie
```

```
        SET salary = salary * 1.05
        WHERE employee_id = emp_id;

        SELECT salary
        INTO emp_salary
        FROM empl_copie
        WHERE employee_id = emp_id;

        DBMS_OUTPUT.PUT_LINE('  Salariul nou: ' || emp_salary);
    END LOOP;
END;
/
```

Worksheet | Query Builder

```
        DBMS_OUTPUT.PUT_LINE('');
    END LOOP;
END;
/

-- 1

-- fac o copie a tabelei employees pt a putea face modificari pe ea
DROP TABLE empl_copie;
CREATE TABLE empl_copie AS (SELECT * FROM employees);
COMMIT;

DECLARE
    -- Colecţie cu idurile celor care nu castiga comision
    TYPE emp_ids_vector IS VARRAY(5)
        OF empl_copie.employee_id%TYPE NOT NULL;
    emps emp_ids_vector := emp_ids_vector();
    emp_id empl_copie.employee_id%TYPE;
    emp_salary empl_copie.salary%TYPE;
BEGIN
```

Script Output | Task completed in 0.137 seconds

```
*Action:
Actualizez salariatul cu id ul1000
  Salariu vechi: 1000
  Salariul nou: 1050
Actualizez salariatul cu id ul132
  Salariu vechi: 2100
  Salariul nou: 2205
Actualizez salariatul cu id ul128
  Salariu vechi: 2200
  Salariul nou: 2310
Actualizez salariatul cu id ul136
  Salariu vechi: 2200
  Salariul nou: 2310
Actualizez salariatul cu id ul127
  Salariu vechi: 2400
  Salariul nou: 2520


PL/SQL procedure successfully completed.
```

PLSQL2 EX2

-- 2

-- VARRAY

```
-- Creez tabelul care reține excursiile
DROP TABLE excursie;

CREATE OR REPLACE TYPE tip_orase_1 AS VARRAY(50) OF VARCHAR2(10);
/

CREATE TABLE excursie_1 (
   cod_excursie NUMBER(4) PRIMARY KEY,
   denumire VARCHAR(20) NOT NULL,
   orase tip_orase_1 NOT NULL,
   status VARCHAR(20) NOT NULL
);

-- Inserare date
DELETE FROM excursie_1;
INSERT ALL
   INTO excursie_1 (cod_excursie, denumire, orase,status)
   VALUES (1, 'Romania', tip_orase_1('Bucuresti', 'Iasi'),'disponibila')
   INTO excursie_1 (cod_excursie, denumire, orase,status)
   VALUES (2, 'Spania', tip_orase_1('Barcelona'),'disponibila')
   INTO excursie_1 (cod_excursie, denumire, orase,status)
   VALUES (3, 'Bulgaria', tip_orase_1('Sofia', 'Ruse'),'disponibila')
   INTO excursie_1 (cod_excursie, denumire, orase,status)
   VALUES (5, 'Romania', tip_orase_1('Sighisoara'),'disponibila')
   INTO excursie_1 (cod_excursie, denumire, orase,status)
   VALUES (6, 'Franta', tip_orase_1('Paris'),'disponibila')
   INTO excursie_1 (cod_excursie, denumire, orase,status)
   VALUES (10, 'Italia', tip_orase_1('Roma'),'disponibila')
   INTO excursie_1 (cod_excursie, denumire, orase,status)
   VALUES (11, 'Portugalia', tip_orase_1('Madeira'),'anulata')
SELECT * FROM dual;

COMMIT;
/
SELECT * FROM excursie_1;
/
set serveroutput on;
DECLARE
   v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
   v_orase tip_orase_1;
BEGIN
   SELECT orase INTO v_orase
   FROM excursie_1
   WHERE cod_excursie = v_cod_excursie;

   -- Adaug orașul nou la finalul listei
   v_orase.EXTEND;
   v_orase(v_orase.COUNT) := &oras;
```

```
    UPDATE excursie_1
    SET orase = v_orase
    WHERE cod_excursie = v_cod_excursie;
END;
/

SELECT * FROM excursie_1;
/
```

Worksheet | Query Builder

```
SELECT * FROM dual;

COMMIT;
/
SELECT * FROM excursie_1;
/
set serveroutput on;
DECLARE
    v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
    v_orase tip_orase_1;
BEGIN
    SELECT orase INTO v_orase
    FROM excursie_1
    WHERE cod_excursie = v_cod_excursie;

    -- Adaug orașul nou la finalul listei
    v_orase.EXTEND;
    v_orase(v_orase.COUNT) := &oras;

    UPDATE excursie_1
    SET orase = v_orase
    WHERE cod_excursie = v_cod_excursie;
END;
/

SELECT * FROM excursie_1;
/

-- Adaug un oraș după primul element din lista
DECLARE
```

Script Output | Query Result

SQL | All Rows Fetched: 7 in 0.028 seconds

	COD_EXCURSIE	DENUMIRE	ORASE	STATUS
1	1	Romania	BDSA_DINUC.TIP_ORASE_1('Bucuresti', 'Iasi', 'Pitesti')	disponibila
2	2	Spania	BDSA_DINUC.TIP_ORASE_1('Barcelona')	disponibila
3	3	Bulgaria	BDSA_DINUC.TIP_ORASE_1('Sofia', 'Ruse')	disponibila
4	5	Romania	BDSA_DINUC.TIP_ORASE_1('Sighisoara')	disponibila
5	6	Franta	BDSA_DINUC.TIP_ORASE_1('Paris')	disponibila
6	10	Italia	BDSA_DINUC.TIP_ORASE_1('Roma')	disponibila
7	11	Portugalia	BDSA_DINUC.TIP_ORASE_1('Madeira')	anulata

```
-- Adaug un oraș după primul element din lista
DECLARE
--codul introdus de la tastatura
    v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
    v_orase tip_orase_1;
```

```
BEGIN
   SELECT orase INTO v_orase
   FROM excursie_1
   WHERE cod_excursie = v_cod_excursie;

   -- Adaug orasul după primul element din vector.
   v_orase.EXTEND;
   FOR i IN REVERSE v_orase.NEXT(2)..v_orase.LAST LOOP
      v_orase(i) := v_orase(i - 1);
   END LOOP;
   v_orase(2) := &oras;
--update dupa ce am adaugat orasul
   UPDATE excursie_1
   SET orase = v_orase
   WHERE cod_excursie = v_cod_excursie;
END;
/

SELECT * FROM excursie_1;
/
```

Worksheet | Query Builder

```
SELECT * FROM excursie_1;
/

-- Adaug un oraș după primul element din lista
DECLARE
--codul introdus de la tastatura
    v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
    v_orase tip_orase_1;
BEGIN
    SELECT orase INTO v_orase
    FROM excursie_1
    WHERE cod_excursie = v_cod_excursie;

    -- Adaug orasul după primul element din vector.
    v_orase.EXTEND;
    FOR i IN REVERSE v_orase.NEXT(2)..v_orase.LAST LOOP
        v_orase(i) := v_orase(i - 1);
    END LOOP;
    v_orase(2) := &oras;
--update dupa ce am adaugat orasul
    UPDATE excursie_1
    SET orase = v_orase
    WHERE cod_excursie = v_cod_excursie;
END;
/

SELECT * FROM excursie_1;
/
```

Script Output | Query Result

SQL | All Rows Fetched: 7 in 0.021 seconds

	COD_EXCURSIE	DENUMIRE	ORASE	STATUS
1	1	Romania	BDSA_DINUC.TIP_ORASE_1('Bucuresti', 'Arad', 'Iasi', 'Pitesti')	disponibila
2	2	Spania	BDSA_DINUC.TIP_ORASE_1('Barcelona')	disponibila
3	3	Bulgaria	BDSA_DINUC.TIP_ORASE_1('Sofia', 'Ruse')	disponibila
4	5	Romania	BDSA_DINUC.TIP_ORASE_1('Sighisoara')	disponibila
5	6	Franta	BDSA_DINUC.TIP_ORASE_1('Paris')	disponibila
6	10	Italia	BDSA_DINUC.TIP_ORASE_1('Roma')	disponibila
7	11	Portugalia	BDSA_DINUC.TIP_ORASE_1('Madeira')	anulata

```
-- inversare ordine
DECLARE
   v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
   v_orase tip_orase;
   v_index1 NUMBER;
   v_index2 NUMBER;
   v_oras_aux VARCHAR(20);
BEGIN
   SELECT orase INTO v_orase
   FROM excursie_1
   WHERE cod_excursie = v_cod_excursie;

   -- caut orasele in vector
   FOR i IN v_orase.FIRST..v_orase.LAST LOOP
      IF v_orase(i) = &nume_oras1 THEN
         v_index1 := i;
```

```
        ELSIF v_orase(i) = &nume_oras2 THEN
            v_index2 := i;
        END IF;
    END LOOP;

    -- Interschimb
    v_oras_aux := v_orase(v_index1);
    v_orase(v_index1) := v_orase(v_index2);
    v_orase(v_index2) := v_oras_aux;
--update dupa schimbare
    UPDATE excursie
    SET orase = v_orase
    WHERE cod_excursie = v_cod_excursie;
END;
/

SELECT * FROM excursie_1;
/

-- stergere oras dupa nume dat de la tastatura
DECLARE
    v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
    v_orase tip_orase_1;
    v_index NUMBER := NULL;
BEGIN
    SELECT orase INTO v_orase
    FROM excursie_1
    WHERE cod_excursie = v_cod_excursie;

    -- Caut orașul în vector
    FOR i IN v_orase.FIRST..v_orase.LAST LOOP
        IF v_orase(i) = &nume_oras THEN
            v_index := i;
        END IF;
    END LOOP;

    -- Șterg orașul
    FOR i IN (v_index + 1)..v_orase.COUNT LOOP
        v_orase(i - 1) := v_orase(i);
    END LOOP;

    v_orase.TRIM;

    UPDATE excursie_1
    SET orase = v_orase
    WHERE cod_excursie = v_cod_excursie;
END;
/
```

SELECT * FROM excursie_1;
/

Worksheet | Query Builder

```
/

SELECT * FROM excursie_1;
/


-- stergere oras dupa nume dat de la tastatura
DECLARE
    v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
    v_orase tip_orase_1;
    v_index NUMBER := NULL;
BEGIN
    SELECT orase INTO v_orase
    FROM excursie_1
    WHERE cod_excursie = v_cod_excursie;

    -- Caut orașul în vector
    FOR i IN v_orase.FIRST..v_orase.LAST LOOP
        IF v_orase(i) = &nume_oras THEN
            v_index := i;
        END IF;
    END LOOP;

    -- Șterg orașul
    FOR i IN (v_index + 1)..v_orase.COUNT LOOP
        v_orase(i - 1) := v_orase(i);
    END LOOP;

    v_orase.TRIM;
```

Script Output | Query Result

SQL | All Rows Fetched: 7 in 0.02 seconds

	COD_EXCURSIE	DENUMIRE	ORASE	STATUS
1	1	Romania	BDSA_DINUC.TIP_ORASE_1('Bucuresti', 'Arad', 'Iasi')	disponibila
2	2	Spania	BDSA_DINUC.TIP_ORASE_1('Barcelona')	disponibila
3	3	Bulgaria	BDSA_DINUC.TIP_ORASE_1('Sofia', 'Ruse')	disponibila
4	5	Romania	BDSA_DINUC.TIP_ORASE_1('Sighisoara')	disponibila
5	6	Franta	BDSA_DINUC.TIP_ORASE_1('Paris')	disponibila
6	10	Italia	BDSA_DINUC.TIP_ORASE_1('Roma')	disponibila
7	11	Portugalia	BDSA_DINUC.TIP_ORASE_1('Madeira')	anulata

```
-- afisare detalii
DECLARE
   v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
   v_orase tip_orase;
BEGIN
   SELECT orase INTO v_orase
   FROM excursie_1
   WHERE cod_excursie = v_cod_excursie;
--lista
```

```
    DBMS_OUTPUT.PUT_LINE('Nr. orase: ' || v_orase.COUNT);

    FOR i IN v_orase.FIRST..v_orase.LAST LOOP
        DBMS_OUTPUT.PUT_LINE('Oras #' || i || ': ' || v_orase(i));
    END LOOP;
END;
/
-- Afisare lista de orașe pentru fiecare excursie, pe rând
DECLARE
    TYPE tip_coduri_excursii IS TABLE OF excursie.cod_excursie%TYPE;
    v_coduri_excursii tip_coduri_excursii;
    v_orase tip_orase;
BEGIN
    FOR e IN (SELECT cod_excursie, orase FROM excursie) LOOP
        DBMS_OUTPUT.PUT_LINE('Excursie #' || e.cod_excursie);

        FOR j IN 1..(e.orase.COUNT) LOOP
            DBMS_OUTPUT.PUT_LINE('  Oras #' || j || ': ' || e.orase(j));
        END LOOP;
    END LOOP;
END;
/
```

```
-- Afisare lista de orașe pentru fiecare excursie, pe rând
DECLARE
    TYPE tip_coduri_excursii IS TABLE OF excursie_1.cod_excursie%TYPE;
    v_coduri_excursii tip_coduri_excursii;
    v_orase tip_orase_1;
BEGIN
    FOR e IN (SELECT cod_excursie, orase FROM excursie_1) LOOP
        DBMS_OUTPUT.PUT_LINE('Excursie ' || e.cod_excursie);

        FOR j IN 1..(e.orase.COUNT) LOOP
            DBMS_OUTPUT.PUT_LINE('  Oras ' || j || ': ' || e.orase(j));
        END LOOP;
    END LOOP;
END;
/

-- Anulare excursie cu cele mai putine orașe vizitate
```

Script Output | Query Result

Task completed in 0.15 seconds

```
Excursie 1
  Oras 1: Bucuresti
  Oras 2: Arad
  Oras 3: Iasi
Excursie 2
  Oras 1: Barcelona
Excursie 3
  Oras 1: Sofia
  Oras 2: Ruse
Excursie 5
```

```
-- Anulare excursie cu cele mai putine orașe vizitate
DECLARE
```

```
    v_nr_min NUMBER := 99999;
BEGIN
    FOR e IN (SELECT orase FROM excursie_1) LOOP
        IF e.orase.COUNT < v_nr_min THEN
            v_nr_min := e.orase.COUNT;
        END IF;
    END LOOP;

    FOR e IN (SELECT cod_excursie, orase FROM excursie_1) LOOP
        -- Daca are număr minim de orașe
        IF e.orase.COUNT = v_nr_min THEN
            -- anulez excursia
            UPDATE excursie_1
            SET status = 'anulata'
            WHERE cod_excursie = e.cod_excursie;
        END IF;
    END LOOP;
END;
/

SELECT * FROM excursie_1;
/
```

```oracle
-- Anulare excursie cu cele mai putine orașe vizitate
DECLARE
    v_nr_min NUMBER := 99999;
BEGIN
    FOR e IN (SELECT orase FROM excursie_1) LOOP
        IF e.orase.COUNT < v_nr_min THEN
            v_nr_min := e.orase.COUNT;
        END IF;
    END LOOP;

    FOR e IN (SELECT cod_excursie, orase FROM excursie_1) LOOP
        -- Dacă are număr minim de orașe
        IF e.orase.COUNT = v_nr_min THEN
            -- Atunci anulez excursia
            UPDATE excursie_1
            SET status = 'anulata'
            WHERE cod_excursie = e.cod_excursie;
        END IF;
    END LOOP;
END;
/

SELECT * FROM excursie_1;
/

delete from excursie_1 where 1=1;
-- 3
-- Implementare cu tabele imbricate
```

Script Output | Query Result | Query Result 1 | Query Result 2

SQL | All Rows Fetched: 7 in 0.093 seconds

	COD_EXCURSIE	DENUMIRE	ORASE	STATUS
1	1	Romania	BDSA_DINUC.TIP_ORASE_1('Bucuresti', 'Arad', 'Iasi')	disponibila
2	2	Spania	BDSA_DINUC.TIP_ORASE_1('Barcelona')	anulata
3	3	Bulgaria	BDSA_DINUC.TIP_ORASE_1('Sofia', 'Ruse')	disponibila
4	5	Romania	BDSA_DINUC.TIP_ORASE_1('Sighisoara')	anulata
5	6	Franta	BDSA_DINUC.TIP_ORASE_1('Paris')	anulata
6	10	Italia	BDSA_DINUC.TIP_ORASE_1('Roma')	anulata
7	11	Portugalia	BDSA_DINUC.TIP_ORASE_1('Madeira')	anulata

PLSQL2 EX 3

```oracle
CREATE OR REPLACE TYPE tip_orase_1 AS TABLE OF VARCHAR2(10);
/

CREATE TABLE excursie_1 (
   cod_excursie NUMBER(4) PRIMARY KEY,
   denumire VARCHAR2(20)  NOT NULL,
   orase tip_orase,
   status VARCHAR2(20) DEFAULT 'disponibila' NOT NULL
)
NESTED TABLE orase STORE AS excursie_orase;

-- Inserez date
DELETE FROM excursie_1;
INSERT ALL
   INTO excursie_1 (cod_excursie, denumire, orase,status)
   VALUES (1, 'Romania', tip_orase_1('Bucuresti', 'Iasi'),'disponibila')
   INTO excursie_1 (cod_excursie, denumire, orase,status)
   VALUES (2, 'Spania', tip_orase_1('Barcelona'),'disponibila')
   INTO excursie_1 (cod_excursie, denumire, orase,status)
```

```
    VALUES (3, 'Bulgaria', tip_orase_1('Sofia', 'Ruse'),'disponibila')
    INTO excursie_1 (cod_excursie, denumire, orase,status)
    VALUES (5, 'Romania', tip_orase_1('Sighisoara'),'disponibila')
    INTO excursie_1 (cod_excursie, denumire, orase,status)
    VALUES (6, 'Franta', tip_orase_1('Paris'),'disponibila')
    INTO excursie_1 (cod_excursie, denumire, orase,status)
    VALUES (10, 'Italia', tip_orase_1('Roma'),'disponibila')
    INTO excursie_1 (cod_excursie, denumire, orase,status)
    VALUES (11, 'Portugalia', tip_orase_1('Madeira'),'anulata')
SELECT * FROM dual;

COMMIT;
/
set serveroutput on;
DECLARE
    v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
    v_orase tip_orase_1;
BEGIN
    SELECT orase INTO v_orase
    FROM excursie_1
    WHERE cod_excursie = v_cod_excursie;

    -- Adaug orașul nou la finalul listei
    v_orase.EXTEND;
    v_orase(v_orase.COUNT) := &oras;

    UPDATE excursie_1
    SET orase = v_orase
    WHERE cod_excursie = v_cod_excursie;
END;
/

SELECT * FROM excursie_1;
/
```

Worksheet | Query Builder

```
SELECT * FROM dual;

COMMIT;
/
SELECT * FROM excursie_1;
/
set serveroutput on;
DECLARE
    v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
    v_orase tip_orase_1;
BEGIN
    SELECT orase INTO v_orase
    FROM excursie_1
    WHERE cod_excursie = v_cod_excursie;

    -- Adaug orașul nou la finalul listei
    v_orase.EXTEND;
    v_orase(v_orase.COUNT) := &oras;

    UPDATE excursie_1
    SET orase = v_orase
    WHERE cod_excursie = v_cod_excursie;
END;
/

SELECT * FROM excursie_1;
/

-- Adaug un oraș după primul element din lista
DECLARE
```

Script Output | Query Result

SQL | All Rows Fetched: 7 in 0.028 seconds

	COD_EXCURSIE	DENUMIRE	ORASE	STATUS
1	1	Romania	BDSA_DINUC.TIP_ORASE_1('Bucuresti', 'Iasi', 'Pitesti')	disponibila
2	2	Spania	BDSA_DINUC.TIP_ORASE_1('Barcelona')	disponibila
3	3	Bulgaria	BDSA_DINUC.TIP_ORASE_1('Sofia', 'Ruse')	disponibila
4	5	Romania	BDSA_DINUC.TIP_ORASE_1('Sighisoara')	disponibila
5	6	Franta	BDSA_DINUC.TIP_ORASE_1('Paris')	disponibila
6	10	Italia	BDSA_DINUC.TIP_ORASE_1('Roma')	disponibila
7	11	Portugalia	BDSA_DINUC.TIP_ORASE_1('Madeira')	anulata

```
-- Adaug un oraș după primul element din lista
DECLARE
--codul introdus de la tastatura
   v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
   v_orase tip_orase_1;
BEGIN
   SELECT orase INTO v_orase
   FROM excursie_1
   WHERE cod_excursie = v_cod_excursie;

   -- Adaug orasul după primul element din vector.
   v_orase.EXTEND;
   FOR i IN REVERSE v_orase.NEXT(2)..v_orase.LAST LOOP
      v_orase(i) := v_orase(i - 1);
```

```
    END LOOP;
    v_orase(2) := &oras;
--update dupa ce am adaugat orasul
    UPDATE excursie_1
    SET orase = v_orase
    WHERE cod_excursie = v_cod_excursie;
END;
/

SELECT * FROM excursie_1;
/
```

Worksheet | Query Builder

```
SELECT * FROM excursie_1;
/

-- Adaug un oraș după primul element din lista
DECLARE
--codul introdus de la tastatura
    v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
    v_orase tip_orase_1;
BEGIN
    SELECT orase INTO v_orase
    FROM excursie_1
    WHERE cod_excursie = v_cod_excursie;

    -- Adaug orasul după primul element din vector.
    v_orase.EXTEND;
    FOR i IN REVERSE v_orase.NEXT(2)..v_orase.LAST LOOP
        v_orase(i) := v_orase(i - 1);
    END LOOP;
    v_orase(2) := &oras;
--update dupa ce am adaugat orasul
    UPDATE excursie_1
    SET orase = v_orase
    WHERE cod_excursie = v_cod_excursie;
END;
/

SELECT * FROM excursie_1;
/
```

Script Output | Query Result

SQL | All Rows Fetched: 7 in 0.021 seconds

	COD_EXCURSIE	DENUMIRE	ORASE	STATUS
1	1	Romania	BDSA_DINUC.TIP_ORASE_1('Bucuresti', 'Arad', 'Iasi', 'Pitesti')	disponibila
2	2	Spania	BDSA_DINUC.TIP_ORASE_1('Barcelona')	disponibila
3	3	Bulgaria	BDSA_DINUC.TIP_ORASE_1('Sofia', 'Ruse')	disponibila
4	5	Romania	BDSA_DINUC.TIP_ORASE_1('Sighisoara')	disponibila
5	6	Franta	BDSA_DINUC.TIP_ORASE_1('Paris')	disponibila
6	10	Italia	BDSA_DINUC.TIP_ORASE_1('Roma')	disponibila
7	11	Portugalia	BDSA_DINUC.TIP_ORASE_1('Madeira')	anulata

```
-- inversare ordine
DECLARE
    v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
    v_orase tip_orase;
```

```
    v_index1 NUMBER;
    v_index2 NUMBER;
    v_oras_aux VARCHAR(20);
BEGIN
    SELECT orase INTO v_orase
    FROM excursie_1
    WHERE cod_excursie = v_cod_excursie;

    -- caut orasele in vector
    FOR i IN v_orase.FIRST..v_orase.LAST LOOP
        IF v_orase(i) = &nume_oras1 THEN
            v_index1 := i;
        ELSIF v_orase(i) = &nume_oras2 THEN
            v_index2 := i;
        END IF;
    END LOOP;

    -- Interschimb
    v_oras_aux := v_orase(v_index1);
    v_orase(v_index1) := v_orase(v_index2);
    v_orase(v_index2) := v_oras_aux;
--update dupa schimbare
    UPDATE excursie
    SET orase = v_orase
    WHERE cod_excursie = v_cod_excursie;
END;
/

SELECT * FROM excursie_1;
/

-- stergere oras dupa nume dat de la tastatura
DECLARE
    v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
    v_orase tip_orase_1;
    v_index NUMBER := NULL;
BEGIN
    SELECT orase INTO v_orase
    FROM excursie_1
    WHERE cod_excursie = v_cod_excursie;

    -- Caut orașul în vector
    FOR i IN v_orase.FIRST..v_orase.LAST LOOP
        IF v_orase(i) = &nume_oras THEN
            v_index := i;
        END IF;
    END LOOP;

    -- Șterg orașul
```

```
    FOR i IN (v_index + 1)..v_orase.COUNT LOOP
        v_orase(i - 1) := v_orase(i);
    END LOOP;

    v_orase.TRIM;

    UPDATE excursie_1
    SET orase = v_orase
    WHERE cod_excursie = v_cod_excursie;
END;
/

SELECT * FROM excursie_1;
/
```

Worksheet | Query Builder

```
/

SELECT * FROM excursie_1;
/


-- stergere oras dupa nume dat de la tastatura
DECLARE
    v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
    v_orase tip_orase_1;
    v_index NUMBER := NULL;
BEGIN
    SELECT orase INTO v_orase
    FROM excursie_1
    WHERE cod_excursie = v_cod_excursie;

    -- Caut orașul în vector
    FOR i IN v_orase.FIRST..v_orase.LAST LOOP
        IF v_orase(i) = &nume_oras THEN
            v_index := i;
        END IF;
    END LOOP;

    -- Șterg orașul
    FOR i IN (v_index + 1)..v_orase.COUNT LOOP
        v_orase(i - 1) := v_orase(i);
    END LOOP;

    v_orase.TRIM;
```

Script Output | Query Result

SQL | All Rows Fetched: 7 in 0.02 seconds

	COD_EXCURSIE	DENUMIRE	ORASE	STATUS
1	1	Romania	BDSA_DINUC.TIP_ORASE_1('Bucuresti', 'Arad', 'Iasi')	disponibila
2	2	Spania	BDSA_DINUC.TIP_ORASE_1('Barcelona')	disponibila
3	3	Bulgaria	BDSA_DINUC.TIP_ORASE_1('Sofia', 'Ruse')	disponibila
4	5	Romania	BDSA_DINUC.TIP_ORASE_1('Sighisoara')	disponibila
5	6	Franta	BDSA_DINUC.TIP_ORASE_1('Paris')	disponibila
6	10	Italia	BDSA_DINUC.TIP_ORASE_1('Roma')	disponibila
7	11	Portugalia	BDSA_DINUC.TIP_ORASE_1('Madeira')	anulata

```
-- afisare detalii
DECLARE
   v_cod_excursie CONSTANT NUMBER NOT NULL := &cod;
   v_orase tip_orase;
BEGIN
   SELECT orase INTO v_orase
   FROM excursie_1
   WHERE cod_excursie = v_cod_excursie;
--lista
   DBMS_OUTPUT.PUT_LINE('Nr. orase: ' || v_orase.COUNT);

   FOR i IN v_orase.FIRST..v_orase.LAST LOOP
```

```
        DBMS_OUTPUT.PUT_LINE('Oras #' || i || ': ' || v_orase(i));
    END LOOP;
END;
/
-- Afisare lista de orașe pentru fiecare excursie, pe rând
DECLARE
    TYPE tip_coduri_excursii IS TABLE OF excursie.cod_excursie%TYPE;
    v_coduri_excursii tip_coduri_excursii;
    v_orase tip_orase;
BEGIN
    FOR e IN (SELECT cod_excursie, orase FROM excursie) LOOP
        DBMS_OUTPUT.PUT_LINE('Excursie #' || e.cod_excursie);

        FOR j IN 1..(e.orase.COUNT) LOOP
            DBMS_OUTPUT.PUT_LINE('  Oras #' || j || ': ' || e.orase(j));
        END LOOP;
    END LOOP;
END;
/
```

```
-- Afisare lista de orașe pentru fiecare excursie, pe rând
DECLARE
    TYPE tip_coduri_excursii IS TABLE OF excursie_1.cod_excursie%TYPE;
    v_coduri_excursii tip_coduri_excursii;
    v_orase tip_orase_1;
BEGIN
    FOR e IN (SELECT cod_excursie, orase FROM excursie_1) LOOP
        DBMS_OUTPUT.PUT_LINE('Excursie ' || e.cod_excursie);

        FOR j IN 1..(e.orase.COUNT) LOOP
            DBMS_OUTPUT.PUT_LINE('  Oras ' || j || ': ' || e.orase(j));
        END LOOP;
    END LOOP;
END;
/

-- Anulare excursie cu cele mai putine orașe vizitate
```

Script Output | Query Result

Task completed in 0.15 seconds

```
Excursie 1
  Oras 1: Bucuresti
  Oras 2: Arad
  Oras 3: Iasi
Excursie 2
  Oras 1: Barcelona
Excursie 3
  Oras 1: Sofia
  Oras 2: Ruse
Excursie 5
```

```
-- Anulare excursie cu cele mai putine orașe vizitate
DECLARE
    v_nr_min NUMBER := 99999;
BEGIN
    FOR e IN (SELECT orase FROM excursie_1) LOOP
```

```
        IF e.orase.COUNT < v_nr_min THEN
            v_nr_min := e.orase.COUNT;
        END IF;
    END LOOP;

    FOR e IN (SELECT cod_excursie, orase FROM excursie_1) LOOP
        -- Daca are număr minim de orașe
        IF e.orase.COUNT = v_nr_min THEN
            -- anulez excursia
            UPDATE excursie_1
            SET status = 'anulata'
            WHERE cod_excursie = e.cod_excursie;
        END IF;
    END LOOP;
END;
/

SELECT * FROM excursie_1;
/
```

PLSQL 3 EX 1 SI EX 2

```
SET VERIFY OFF;
set serveroutput on
-- 1

DECLARE
    v_num_ang NUMBER;
    v_sal_lunar NUMBER;
    v_sal_mediu NUMBER;
    v_contor_ang NUMBER;

    v_nr_ang NUMBER;
    v_sal_total NUMBER;
    v_sal_total_med NUMBER;
BEGIN
    -- Parcurgere joburi folosind un ciclu cursor
    FOR i IN (SELECT job_id, job_title FROM jobs)
    LOOP
        DBMS_OUTPUT.PUT_LINE('Job: ' || i.job_title);
--Extragere despre jobul curent (nr, salariu lunar, salariu mediu)
        SELECT COUNT(1), SUM(salary), AVG(salary)
        INTO v_num_ang, v_sal_lunar, v_sal_mediu
        FROM employees
        WHERE job_id = i.job_id;
        IF v_num_ang = 0 THEN
        --afisare si tratarea cazului in care nu exista angajati cu jobul precizat
            DBMS_OUTPUT.PUT_LINE('Nu avem angajati cu acest job');
        ELSE
```

```
            DBMS_OUTPUT.PUT_LINE('- Numar angajati: ' || v_num_ang);
            DBMS_OUTPUT.PUT_LINE('- Salariu lunar total: ' || v_sal_lunar);
            DBMS_OUTPUT.PUT_LINE('- Salariu mediu: ' || v_sal_mediu);
        END IF;

        -- Afisare angajati
        v_contor_ang := 1;
        FOR j IN (SELECT first_name, last_name, salary
                  FROM employees
                  WHERE job_id = i.job_id)
        LOOP
            DBMS_OUTPUT.PUT_LINE(
                '    Angajatul ' || v_contor_ang || ': '
                || j.first_name || ' ' || j.last_name
                || ' are salariul ' || j.salary
            );

            -- folosesc cursor pentru a numara angajatii
            v_contor_ang := v_contor_ang + 1;
        END LOOP;

    END LOOP;
    SELECT
    --selectez detalii pentru toti angajatii--
        COUNT(1),
        SUM(salary),
        Round(AVG(salary))
    INTO
        v_nr_ang,
        v_sal_total,
        v_sal_total_med
    FROM employees;

    DBMS_OUTPUT.PUT_LINE('Numar total angajati: ' || v_nr_ang);
    DBMS_OUTPUT.PUT_LINE('Salariu  total angajati: ' || v_sal_total);
    DBMS_OUTPUT.PUT_LINE('Salariu  mediu angajati: ' || v_sal_total_med);
END;
/
```

0.493 seconds

Worksheet | Query Builder

```
            v_contor_ang := v_contor_ang + 1;
        END LOOP;

    END LOOP;
    SELECT
    --selectez detalii pentru toti angajatii--
        COUNT(1),
        SUM(salary),
        Round(AVG(salary))
    INTO
        v_nr_ang,
        v_sal_total,
        v_sal_total_med
    FROM employees;

    DBMS_OUTPUT.PUT_LINE('Numar total angajati: ' || v_nr_ang);
    DBMS_OUTPUT.PUT_LINE('Salariu  total angajati: ' || v_sal_total);
    DBMS_OUTPUT.PUT_LINE('Salariu  mediu angajati: ' || v_sal_total_med);
END;
/
```

Script Output

Task completed in 0.493 seconds

```
    Angajatul 10: Jennifer Dilly are salariul 3600
    Angajatul 11: Timothy Gates are salariul 2900
    Angajatul 12: Randall Perkins are salariul 2500
    Angajatul 13: Sarah Bell are salariul 4000
    Angajatul 14: Britney Everett are salariul 3900
    Angajatul 15: Samuel McCain are salariul 3200
    Angajatul 16: Vance Jones are salariul 2800
    Angajatul 17: Alana Walsh are salariul 3100
    Angajatul 18: Kevin Feeney are salariul 3000
    Angajatul 19: Donald OConnell are salariul 2600
    Angajatul 20: Douglas Grant are salariul 2600
Job: Programmer
- Numar angajati: 5
- Salariu lunar total: 28800
- Salariu mediu: 5760
    Angajatul 1: Alexander Hunold are salariul 9000
    Angajatul 2: Bruce Ernst are salariul 6000
    Angajatul 3: David Austin are salariul 4800
    Angajatul 4: Valli Pataballa are salariul 4800
    Angajatul 5: Diana Lorentz are salariul 4200
Job: Marketing Manager
```

Worksheet | Query Builder

```
WHERE job_id = i.job_id;
IF v_num_ang = 0 THEN
--afisare si tratarea cazului in care nu exista angajati cu jobul
    DBMS_OUTPUT.PUT_LINE('Nu avem angajati cu acest job');
ELSE
    DBMS_OUTPUT.PUT_LINE('- Numar angajati: ' || v_num_ang);
    DBMS_OUTPUT.PUT_LINE('- Salariu lunar total: ' || v_sal_lunar)
    DBMS_OUTPUT.PUT_LINE('- Salariu mediu: ' || v_sal_mediu);
END IF;

-- Afisare angajati
v_contor_ang := 1;
FOR j IN (SELECT first_name, last_name, salary
          FROM employees
          WHERE job_id = i.job_id)
LOOP
    DBMS_OUTPUT.PUT_LINE(
        '    Angajatul ' || v_contor_ang || ': '
       || j.first_name || ' ' || j.last_name
       || ' are salariul ' || j.salary
```

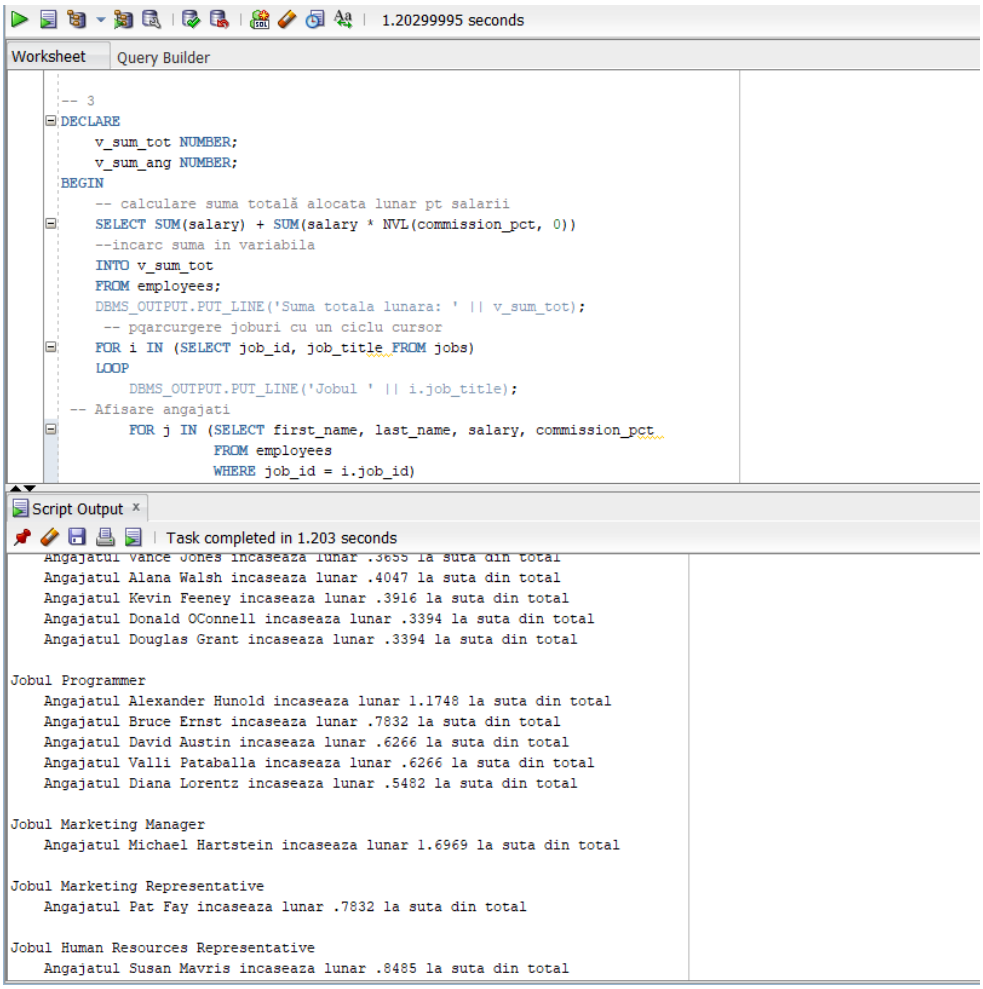
Script Output | Task completed in 0.493 seconds

```
PL/SQL procedure successfully completed.

Job: President
- Numar angajati: 1
- Salariu lunar total: 24000
- Salariu mediu: 24000
    Angajatul 1: Steven King are salariul 24000
Job: Administration Vice President
- Numar angajati: 3
- Salariu lunar total: 35000
- Salariu mediu: 11666.6666666666666666666666666666667
    Angajatul 1: Neena Kochhar are salariul 17000
    Angajatul 2: Lex De Haan are salariul 17000
    Angajatul 3: John John are salariul 1000
Job: Administration Assistant
- Numar angajati: 1
- Salariu lunar total: 4400
- Salariu mediu: 4400
    Angajatul 1: Jennifer Whalen are salariul 4400
Job: Finance Manager
```

PLSQL3 EX 3

```
-- 3
DECLARE
   v_sum_tot                                                    NUMBER;
   v_sum_ang                                                    NUMBER;
BEGIN
   -- calculare suma                                            totală alocata lunar pt
salarii
   SELECT SUM(salary) + SUM(salary * NVL(commission_pct, 0))
   --incarc suma in variabila
   INTO v_sum_tot
   FROM employees;
   DBMS_OUTPUT.PUT_LINE('Suma totala lunara: ' || v_sum_tot);
    -- pqarcurgere joburi cu un ciclu cursor
   FOR i IN (SELECT job_id, job_title FROM jobs)
   LOOP
      DBMS_OUTPUT.PUT_LINE('Jobul ' || i.job_title);
 -- Afisare angajati
      FOR j IN (SELECT first_name, last_name, salary, commission_pct
              FROM employees
              WHERE job_id = i.job_id)
      LOOP
         --calcul suma pt un angajat
         v_sum_ang := j.salary * (1 + NVL(j.commission_pct, 0));
         DBMS_OUTPUT.PUT_LINE(
            '    Angajatul '
         || j.first_name || ' ' || j.last_name
         || ' incaseaza lunar ' ||
         ROUND(v_sum_ang / v_sum_tot * 100, 4)
```

```
            || ' la suta din total'
        );
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('');
  END LOOP;
END
```



PLSQL3 EX 4

```
-- 4
DECLARE
  TYPE t_sal_ang IS RECORD (
    id NUMBER,
    name VARCHAR2(50),
    salary NUMBER
  );
  TYPE t_vec_sal_ang IS VARRAY(5) OF t_sal_ang;
  v_top_ang t_vec_sal_ang;
BEGIN
--parcurg joburile
  FOR i IN (SELECT job_id, job_title FROM jobs)
  LOOP
```

```
        DBMS_OUTPUT.PUT_LINE('Job: ' || i.job_title);

        -- selectez primii 5 angajati in functie de salariu
        SELECT *
        BULK COLLECT INTO v_top_ang
        FROM (
            SELECT
                employee_id,
                first_name || ' ' || last_name,
                salary
            FROM employees
            WHERE job_id = i.job_id
            ORDER BY salary DESC
        )
        WHERE rownum <= 5;
--tratez cazul in care sunt mai putin de 5 angajati pt jobul precizat
        IF v_top_ang.COUNT < 5 THEN
            DBMS_OUTPUT.PUT_LINE('Jobul precizat are mai putin de 5 angajati');
        END IF;
        -- Dacă am macar un angajat -->afisez
        IF v_top_ang.COUNT > 0 THEN
            FOR i IN v_top_ang.FIRST..v_top_ang.LAST LOOP
                DBMS_OUTPUT.PUT_LINE(v_top_ang(i).name);
            END LOOP;
        END IF;

        DBMS_OUTPUT.PUT_LINE('');
    END LOOP;
END;
/
```

PLSQL 3 EX 5

-- 5

0.36899999 seconds

Worksheet Query Builder

```
/

-- 4
DECLARE
    TYPE t_sal_ang IS RECORD (
        id NUMBER,
        name VARCHAR2(50),
        salary NUMBER
    );
    TYPE t_vec_sal_ang IS VARRAY(5) OF t_sal_ang;
    v_top_ang t_vec_sal_ang;
BEGIN
--parcurg joburile
    FOR i IN (SELECT job_id, job_title FROM jobs)
    LOOP
        DBMS_OUTPUT.PUT_LINE('Job: ' || i.job_title);

        -- selectez primii 5 angajati in functie de salariu
        SELECT *
        BULK COLLECT INTO v_top_ang
```

Script Output

Task completed in 0.369 seconds

```
Valli Pataballa
Diana Lorentz

Job: Marketing Manager
Jobul precizat are mai putin de 5 angajati
Michael Hartstein

Job: Marketing Representative
Jobul precizat are mai putin de 5 angajati
Pat Fay

Job: Human Resources Representative
Jobul precizat are mai putin de 5 angajati
Susan Mavris

Job: Public Relations Representative
Jobul precizat are mai putin de 5 angajati
Hermann Baer
```

```
DECLARE
   -- incarc in variabila v_nr_ang nr de salarii distincte calculate
   v_nr_ang NUMBER;
   v_ult_sal NUMBER;
BEGIN
   FOR i IN (SELECT job_id, job_title FROM jobs)
   LOOP
      DBMS_OUTPUT.PUT_LINE('Job: ' || i.job_title);

      v_nr_ang := 0;
      v_ult_sal := -1;
 -- Parcurgere  angajați in functie de salariu
      FOR e IN (
            SELECT
```

```
            employee_id,
            first_name || ' ' || last_name AS name,
            salary
          FROM employees
          WHERE job_id = i.job_id
          ORDER BY salary DESC
        )
      LOOP
        -- ies din loop cand am afisat primii 5 angajati cu cele mai mari salarii
        EXIT WHEN v_nr_ang >= 5;
        IF e.salary != v_ult_sal THEN
          v_nr_ang := v_nr_ang + 1;
        END IF;

        DBMS_OUTPUT.PUT_LINE('Locul: ' || v_nr_ang || ': '
          || e.name || ' cu ' || e.salary);

        v_ult_sal := e.salary;
      END LOOP;

      DBMS_OUTPUT.PUT_LINE('');
    END LOOP;
END;
/
```

0.235 seconds

Worksheet | Query Builder

```
/
-- 5
DECLARE
    -- incarc in variabila v_nr_ang nr de salarii distincte calculate
    v_nr_ang NUMBER;
    v_ult_sal NUMBER;
BEGIN
    FOR i IN (SELECT job_id, job_title FROM jobs)
    LOOP
        DBMS_OUTPUT.PUT_LINE('Job: ' || i.job_title);

        v_nr_ang := 0;
        v_ult_sal := -1;
-- Parcurgere  angajaţi in functie de salariu
        FOR e IN (
                SELECT
                    employee_id,
                    first_name || ' ' || last_name AS name,
                    salary
                FROM employees
```

Script Output

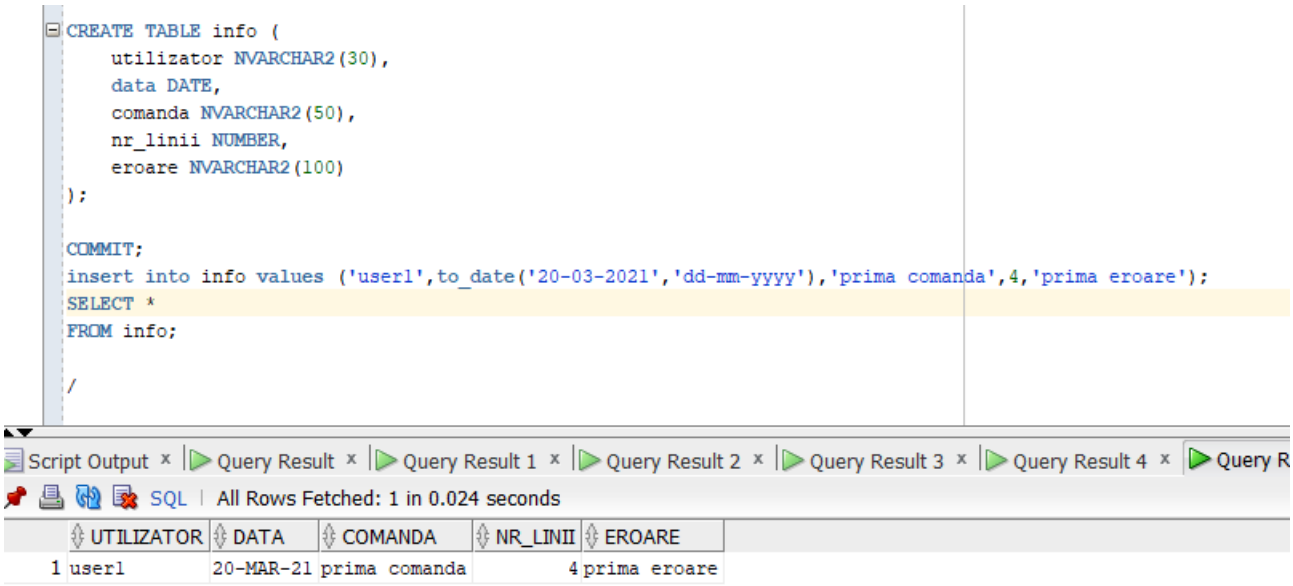
Task completed in 0.235 seconds

```
Locul: 4: Britney Everett cu 3900
Locul: 5: Kelly Chung cu 3800

Job: Programmer
Locul: 1: Alexander Hunold cu 9000
Locul: 2: Bruce Ernst cu 6000
Locul: 3: David Austin cu 4800
Locul: 3: Valli Pataballa cu 4800
Locul: 4: Diana Lorentz cu 4200

Job: Marketing Manager
Locul: 1: Michael Hartstein cu 13000

Job: Marketing Representative
Locul: 1: Pat Fay cu 6000

Job: Human Resources Representative
Locul: 1: Susan Mavris cu 6500

Job: Public Relations Representative
Locul: 1: Hermann Baer cu 10000
```

PLSQL4 EX 1

-- 1

```
DROP TABLE info;
--creare tabela info cu atributele mentionate
CREATE TABLE info (
   utilizator NVARCHAR2(30),
   data DATE,
   comanda NVARCHAR2(50),
   nr_linii NUMBER,
   eroare NVARCHAR2(100)
);

COMMIT;
--inserare in tabela
insert into info values ('user1',to_date('20-03-2021','dd-mm-yyyy'),'prima comanda',4,'prima eroare');
SELECT *
FROM info;

/
```



PLSQL4 EX2

```
--f2--
-- 2
set serveroutput on
CREATE OR REPLACE FUNCTION f2 (
--daun numele angajatului prin parametru
   v_nume employees.last_name%TYPE
)
RETURN NUMBER IS
```

```
   salariu employees.salary%TYPE;
BEGIN
--selectez salariul pentru angajatul dat prin parametru
   SELECT
      salary
   INTO salariu
   FROM
      employees
   WHERE
      last_name = v_nume;

   -- daca s-a gasit doar un angajat cu numele dat -->inserez
   INSERT INTO info VALUES (USER, SYSDATE, 'F2', 1, NULL);
   COMMIT;
   RETURN salariu;
EXCEPTION
   WHEN no_data_found THEN
   --nu s-a gasit angajatul--
      INSERT INTO info VALUES (USER, SYSDATE, 'F2', 0, 'Nu exista angajatul');
      COMMIT;
       raise_application_error(-20000, 'Nu exista angajati cu numele dat');
   WHEN too_many_rows THEN
   --s-au gasit mai multi angajati cu numele dat
      INSERT INTO info VALUES (USER, SYSDATE, 'F2', 0, 'Exista mai multi angajati cu numele
dat');
      COMMIT;
       raise_application_error(-20001, 'Exista mai multi angajati cu numele dat');
   WHEN OTHERS THEN
      INSERT INTO info VALUES (USER, SYSDATE, 'F2', 0, 'Alta eroare');
      COMMIT;

      raise_application_error(-20002, 'Alta eroare');
END f2;
/

DECLARE
   v_salary NUMBER;
BEGIN
   v_salary := f2('Bell2');
END;
/
```

Worksheet | Query Builder

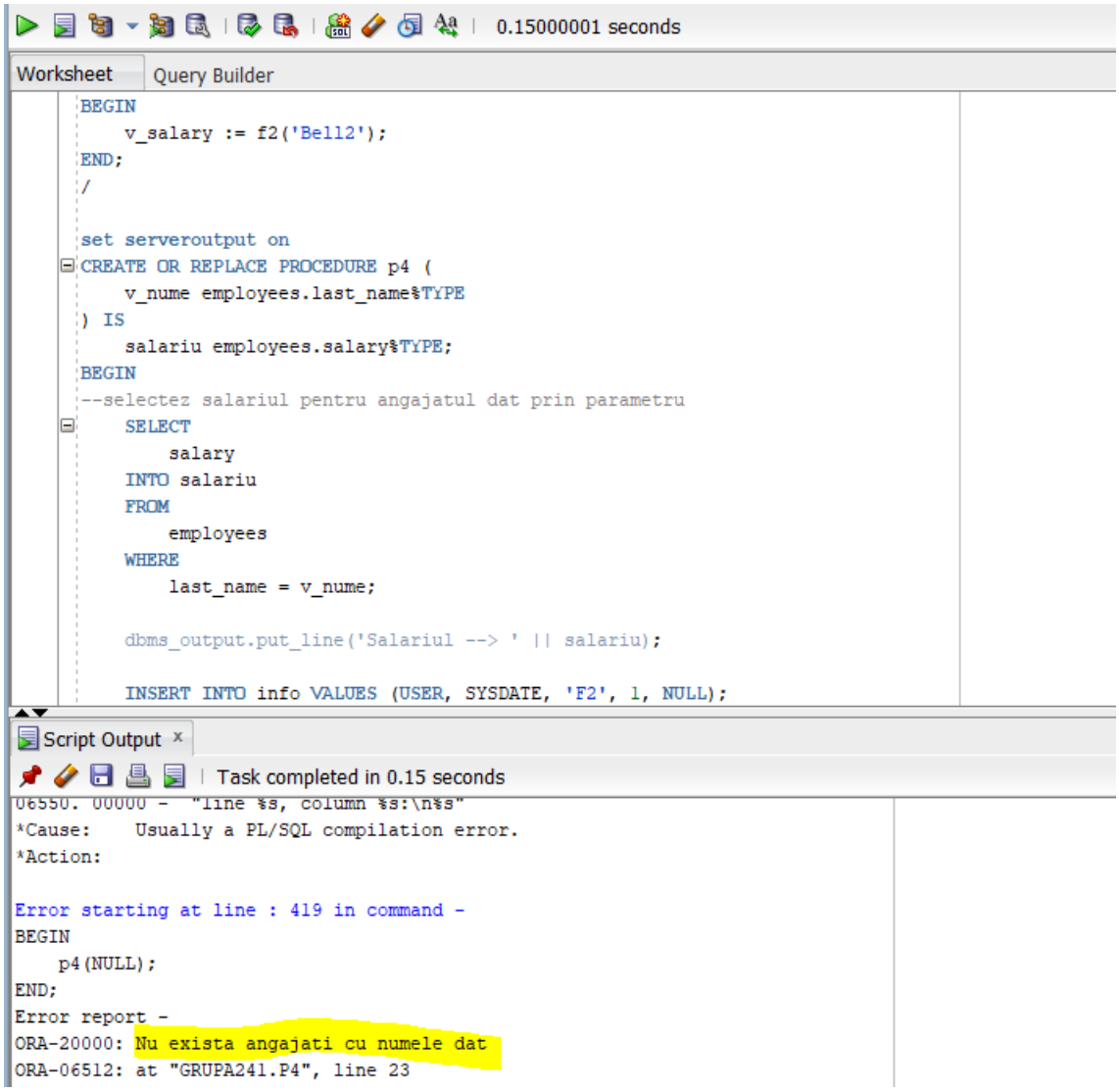
```
        COMMIT;
         raise_application_error(-20000, 'Nu exista angajati cu numele dat');
    WHEN too_many_rows THEN
    --s-au gasit mai multi angajati cu numele dat
        INSERT INTO info VALUES (USER, SYSDATE, 'F2', 0, 'Exista mai multi angajati cu numele dat');
        COMMIT;
         raise_application_error(-20001, 'Exista mai multi angajati cu numele dat');
    WHEN OTHERS THEN
        INSERT INTO info VALUES (USER, SYSDATE, 'F2', 0, 'Alta eroare');
        COMMIT;

        raise_application_error(-20002, 'Alta eroare');
END f2;
/

DECLARE
    v_salary NUMBER;
BEGIN
    v_salary := f2('Bell2');
END;
/
```

Script Output | Task completed in 0.167 seconds

```
Error starting at line : 375 in command -
DECLARE
    v_salary NUMBER;
BEGIN
    v_salary := f2('Bell2');
END;
Error report -
ORA-20000: Nu exista angajati cu numele dat
ORA-06512: at "GRUPA241.F2", line 26
ORA-06512: at line 4
```

```
--p4--
set serveroutput on
CREATE OR REPLACE PROCEDURE p4 (
   v_nume employees.last_name%TYPE
) IS
   salariu employees.salary%TYPE;
BEGIN
--selectez salariul pentru angajatul dat prin parametru
   SELECT
      salary
   INTO salariu
   FROM
      employees
   WHERE
      last_name = v_nume;

   dbms_output.put_line('Salariul --> ' || salariu);
```

```
   INSERT INTO info VALUES (USER, SYSDATE, 'F2', 1, NULL);
   COMMIT;
EXCEPTION
   WHEN no_data_found THEN
      INSERT INTO info VALUES (USER, SYSDATE, 'F2', 0, 'Nu exista angajatul');
      COMMIT;
      raise_application_error(-20000, 'Nu exista angajati cu numele dat');
   WHEN too_many_rows THEN
      INSERT INTO info VALUES (USER, SYSDATE, 'F2', 0, 'Exista mai multi angajati cu numele
dat');
      COMMIT;

      raise_application_error(-20001, 'Exista mai multi angajati cu numele dat');
   WHEN OTHERS THEN
      INSERT INTO info VALUES (USER, SYSDATE, 'F2', 0, 'Alta eroare');
      COMMIT;

      raise_application_error(-20002, 'Alta eroare');
END p4;
/

BEGIN
   p4(NULL);
END;
/
```



PLSQL4 EX4

```
-- 4
set serveroutput on
CREATE OR REPLACE PROCEDURE marire_sal (
   v_manager_id employees.manager_id%TYPE
) IS
--index by table
   TYPE t_ids IS TABLE OF employees.employee_id%TYPE;
   v_ids t_ids;
BEGIN
   -- Selectez angajatii care lucrează direct sau indirect pt managerul dat prin parametru
   SELECT employee_id
   BULK COLLECT INTO v_ids
   FROM employees
   START WITH employee_id = v_manager_id
   CONNECT BY manager_id = PRIOR employee_id;
   -- tratez cazul in care nu exista manager cu codul dat
   IF v_ids.COUNT = 0 THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Nu există un manager cu codul dat');
        RETURN;
    END IF;
    --daca exista-->
    -- Măresc salariile
    FOR i IN v_ids.FIRST..v_ids.LAST
    LOOP
        UPDATE employees
        SET salary = salary * 1.1
        WHERE employee_id = v_ids(i);
    END LOOP;
END;
/

set serveroutput on
BEGIN

    marire_sal(100);
END;
/
select * from employees where manager_id=100;

SELECT salary
FROM employees;
ROLLBACK;
```

INAINTE DE ROLLBACK:

Worksheet | Query Builder

```
    BULK COLLECT INTO v_ids
    FROM employees
    START WITH employee_id = v_manager_id
    CONNECT BY manager_id = PRIOR employee_id;
    -- tratez cazul in care nu exista manager cu codul dat
    IF v_ids.COUNT = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Nu există un manager cu codul dat');
        RETURN;
    END IF;
    --daca exista-->
    -- Măresc salariile
    FOR i IN v_ids.FIRST..v_ids.LAST
    LOOP
        UPDATE employees
        SET salary = salary * 1.1
        WHERE employee_id = v_ids(i);
    END LOOP;
END;
/

set serveroutput on
BEGIN
```

Script Output | Query Result

SQL | All Rows Fetched: 14 in 0.025 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	20570	(null)	100	90
2	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	20570	(null)	100	90
3	114	Den	Raphaely	DRAPHEAL	515.127.4561	07-DEC-94	PU_MAN	13310	(null)	100	30
4	120	Matthew	Weiss	MWEISS	650.123.1234	18-JUL-96	ST_MAN	9680	(null)	100	50
5	121	Adam	Fripp	AFRIPP	650.123.2234	10-APR-97	ST_MAN	9922	(null)	100	50
6	122	Payam	Kaufling	PKAUFLIN	650.123.3234	01-MAY-95	ST_MAN	9559	(null)	100	50
7	123	Shanta	Vollman	SVOLLMAN	650.123.4234	10-OCT-97	ST_MAN	7865	(null)	100	50
8	124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	7018	(null)	100	50
9	145	John	Russell	JRUSSEL	011.44.1344.429268	01-OCT-96	SA_MAN	16940	0.4	100	80
10	146	Karen	Partners	KPARTNER	011.44.1344.467268	05-JAN-97	SA_MAN	16335	0.3	100	80
11	147	Alberto	Errazuriz	AERRAZUR	011.44.1344.429278	10-MAR-97	SA_MAN	14520	0.3	100	80
12	148	Gerald	Cambrault	GCAMBRAU	011.44.1344.619268	15-OCT-99	SA_MAN	13310	0.3	100	80
13	149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00	SA_MAN	12705	0.2	100	80
14	201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	15730	(null)	100	20

DUPA ROLLBACK:

Welcome Page | proiect1 | proiect1~1 | grupa241 | EMPLOYEES

Worksheet | Query Builder

```
    LOOP
        UPDATE employees
        SET salary = salary * 1.1
        WHERE employee_id = v_ids(i);
    END LOOP;
END;
/

set serveroutput on
BEGIN

    marire_sal(100);
END;
/
select * from employees where manager_id=100;

SELECT salary
FROM employees;

ROLLBACK;
```

Script Output | Query Result

SQL | All Rows Fetched: 14 in 0.021 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	18700	(null)	100	90
2	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	18700	(null)	100	90
3	114	Den	Raphaely	DRAPHEAL	515.127.4561	07-DEC-94	PU_MAN	12100	(null)	100	30
4	120	Matthew	Weiss	MWEISS	650.123.1234	18-JUL-96	ST_MAN	8800	(null)	100	50
5	121	Adam	Fripp	AFRIPP	650.123.2234	10-APR-97	ST_MAN	9020	(null)	100	50
6	122	Payam	Kaufling	PKAUFLIN	650.123.3234	01-MAY-95	ST_MAN	8690	(null)	100	50
7	123	Shanta	Vollman	SVOLLMAN	650.123.4234	10-OCT-97	ST_MAN	7150	(null)	100	50
8	124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	6380	(null)	100	50
9	145	John	Russell	JRUSSEL	011.44.1344.429268	01-OCT-96	SA_MAN	15400	0.4	100	80
10	146	Karen	Partners	KPARTNER	011.44.1344.467268	05-JAN-97	SA_MAN	14850	0.3	100	80
11	147	Alberto	Errazuriz	AERRAZUR	011.44.1344.429278	10-MAR-97	SA_MAN	13200	0.3	100	80
12	148	Gerald	Cambrault	GCAMBRAU	011.44.1344.619268	15-OCT-99	SA_MAN	12100	0.3	100	80
13	149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00	SA_MAN	11550	0.2	100	80
14	201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	14300	(null)	100	20

PLSQL 5 EX 1

```
CREATE OR REPLACE PACKAGE pachet_gest_companie
IS
   -- gaseste jobul dat prin parametru
   FUNCTION gas_job(
      nume jobs.job_title%TYPE
   ) RETURN jobs.job_id%TYPE;
-- gaseste angajatul dat prin parametru
   FUNCTION gas_angajat(
      prenume emp.first_name%TYPE,
      nume emp.last_name%TYPE
   ) RETURN emp.employee_id%TYPE;
   --gaseste departamentul dat prin parametru
   FUNCTION gas_departament(
      nume dept.department_name%TYPE
   ) RETURN dept.department_id%TYPE;
   FUNCTION gas_cel_mai_mic_salariu(
      id_dept dept.department_id%TYPE,
      id_job jobs.job_id%TYPE
   ) RETURN emp.salary%TYPE;
   -- comisionul minim pentru jobul si departamentul dat prin parametru--
   --trateaza si exceptia daca nu exista departamentul dat--
   FUNCTION gas_cel_mai_mic_comision(
      id_dept dept.department_id%TYPE,
      id_job jobs.job_id%TYPE
   ) RETURN emp.commission_pct%TYPE;
 -- Adaugă un nou angajat.
   PROCEDURE adauga_ang(
      prenume emp.first_name%TYPE,
      nume emp.last_name%TYPE,
      adresa_email emp.email%TYPE,
      telefon emp.phone_number%TYPE,
      prenume_manager emp.first_name%TYPE,
      nume_manager emp.last_name%TYPE,
      nume_departament dept.department_name%TYPE,
      nume_job jobs.job_title%TYPE
   );

   --muta angajatul dat prin parametru in alt departament
   PROCEDURE muta_ang(
      prenume emp.first_name%TYPE,
      nume emp.last_name%TYPE,
      nume_departament dept.department_name%TYPE,
      nume_job jobs.job_title%TYPE,
      prenume_manager emp.first_name%TYPE,
      nume_manager emp.last_name%TYPE
   );
   --calculeaza nr de subalterni ai unui manager
```

```
    FUNCTION nr_subalterni(
        prenume emp.first_name%TYPE,
        nume emp.last_name%TYPE
    ) RETURN NUMBER;
 -- cursor care afiseaza angajatii de pe un anumit job
    CURSOR angajati_job (id_job jobs.job_id%TYPE)
    RETURN emp%ROWTYPE
    IS
    SELECT * FROM emp
    WHERE job_id = id_job;
  -- Cursor care returnează toate joburile din companie.
    CURSOR toate_joburile
    RETURN jobs%ROWTYPE
    IS
    SELECT * FROM jobs;

    PROCEDURE afiseaza_angajati_per_job;
END;
/

CREATE OR REPLACE PACKAGE BODY pachet_gest_companie
IS
    FUNCTION gas_job(
        nume jobs.job_title%TYPE
    ) RETURN jobs.job_id%TYPE
    IS
        v_id jobs.job_id%TYPE;
    BEGIN
        SELECT job_id INTO v_id
        FROM jobs
        WHERE job_title = nume;

        RETURN v_id;

    EXCEPTION
        WHEN no_data_found THEN
            raise_application_error(-20000, 'Nu am gasit jobul ' || nume);
    END;

    FUNCTION gas_angajat(
        prenume emp.first_name%TYPE,
        nume emp.last_name%TYPE
    ) RETURN emp.employee_id%TYPE
    IS
        v_id emp.employee_id%TYPE;
    BEGIN
        SELECT employee_id INTO v_id
        FROM emp
        WHERE first_name = prenume AND last_name = nume;
```

```
    RETURN v_id;

EXCEPTION
    WHEN no_data_found THEN
        raise_application_error(-20000, 'Nu am gasit angajatul '
                || prenume || ' ' || nume);
END;

FUNCTION gas_departament(
    nume dept.department_name%TYPE
) RETURN dept.department_id%TYPE
IS
    v_id dept.department_id%TYPE;
BEGIN
    SELECT department_id INTO v_id
    FROM dept
    WHERE department_name = nume;

    RETURN v_id;

EXCEPTION
    WHEN no_data_found THEN
        raise_application_error(-20000, 'Nu am gasit departamentul ' || nume);
END;

FUNCTION gas_cel_mai_mic_salariu(
    id_dept dept.department_id%TYPE,
    id_job jobs.job_id%TYPE
) RETURN emp.salary%TYPE
IS
    v_salariu emp.salary%TYPE;
BEGIN
    SELECT MIN(salary) INTO v_salariu
    FROM emp
    WHERE department_id = id_dept AND job_id = id_job;

    IF v_salariu IS NULL THEN
        raise_application_error(-20000,
            'Nu exista angajati cu departamentul si jobul dat');
    END IF;

    RETURN v_salariu;
END;

FUNCTION gas_cel_mai_mic_comision(
    id_dept dept.department_id%TYPE,
    id_job jobs.job_id%TYPE
) RETURN emp.commission_pct%TYPE
```

```
IS
   v_com emp.commission_pct%TYPE;
BEGIN
   SELECT MIN(commission_pct) INTO v_com
   FROM emp
   WHERE department_id = id_dept AND job_id = id_job;

   RETURN v_com;
END;


PROCEDURE adauga_ang(
   prenume emp.first_name%TYPE,
   nume emp.last_name%TYPE,
   adresa_email emp.email%TYPE,
   telefon emp.phone_number%TYPE,
   prenume_manager emp.first_name%TYPE,
   nume_manager emp.last_name%TYPE,
   nume_departament dept.department_name%TYPE,
   nume_job jobs.job_title%TYPE
) IS
   v_emp_id emp.employee_id%TYPE;
   v_dept_id dept.department_id%TYPE;
   v_job_id emp.job_id%TYPE;
   v_salariu emp.salary%TYPE;
BEGIN
   v_emp_id := emp_seq.NEXTVAL;
   v_dept_id := gas_departament(nume_departament);
   v_job_id := gas_job(nume_job);
   v_salariu := gas_cel_mai_mic_salariu(v_dept_id, v_job_id);

   INSERT INTO emp (
      employee_id, first_name, last_name,
      email, phone_number, hire_date,
      job_id,
      salary, commission_pct,
      manager_id,
      department_id
   )
   VALUES (
      v_emp_id, prenume, nume,
      adresa_email, telefon, SYSDATE,
      v_job_id,
      v_salariu, 0,
      gas_angajat(prenume_manager, nume_manager),
      v_dept_id
   );
END;
```

```
PROCEDURE muta_ang(
    prenume emp.first_name%TYPE,
    nume emp.last_name%TYPE,
    nume_departament dept.department_name%TYPE,
    nume_job jobs.job_title%TYPE,
    prenume_manager emp.first_name%TYPE,
    nume_manager emp.last_name%TYPE
) IS
    v_ang emp%ROWTYPE;
    v_manager_nou emp.employee_id%TYPE;
    v_dept_nou dept.department_id%TYPE;
    v_job_nou emp.job_id%TYPE;
    v_salariu_nou emp.salary%TYPE;
    v_com_nou emp.commission_pct%TYPE;
BEGIN
    SELECT * INTO v_ang
    FROM emp
    WHERE employee_id = gas_angajat(prenume, nume);

    INSERT INTO job_hist (
        employee_id,
        start_date, end_date,
        job_id, department_id
    )
    VALUES (
        v_ang.employee_id,
        v_ang.hire_date, SYSDATE,
        v_ang.job_id, v_ang.department_id
    );

    v_dept_nou := gas_departament(nume_departament);
    v_job_nou := gas_job(nume_job);
    v_manager_nou := gas_angajat(prenume_manager, nume_manager);
    v_salariu_nou := gas_cel_mai_mic_salariu(v_dept_nou, v_job_nou);
    v_com_nou := gas_cel_mai_mic_comision(v_dept_nou, v_job_nou);

    UPDATE emp
    SET
        department_id = v_dept_nou,
        job_id = v_job_nou,
        manager_id = v_manager_nou,
        salary = v_salariu_nou,
        commission_pct = v_com_nou,
        hire_date = SYSDATE
    WHERE employee_id = v_ang.employee_id;
END;

FUNCTION nr_subalterni(
    prenume emp.first_name%TYPE,
```

```
        nume emp.last_name%TYPE
    ) RETURN NUMBER
    IS
        v_num NUMBER;
    BEGIN
        SELECT COUNT(*) INTO v_num
        FROM employees
        START WITH employee_id = gas_angajat(prenume, nume)
        CONNECT BY PRIOR employee_id = manager_id;

        RETURN v_num;
    END;

    PROCEDURE afiseaza_angajati_per_job
    IS
        v_nume_job jobs.job_title%TYPE;
        v_nume_ang NVARCHAR2(500);
        a_mai_avut_jobul NUMBER;
    BEGIN
        FOR job IN toate_joburile LOOP
            DBMS_OUTPUT.PUT_LINE('Jobul ' || job.job_title || ':');

            FOR emp IN angajati_job(job.job_id) LOOP
                v_nume_ang := emp.first_name || ' ' || emp.last_name;

                SELECT COUNT(*) INTO a_mai_avut_jobul
                FROM job_hist
                WHERE (employee_id = emp.employee_id) AND
                      (job_id = job.job_id);

                DBMS_OUTPUT.PUT_LINE('  ' || v_nume_ang ||
                    ' - ' || a_mai_avut_jobul);
            END LOOP;
        END LOOP;
    END;
END;
/

-- Testez funcțiile din pachet.
BEGIN

    pachet_gest_companie.adauga_ang(
        'Ang1', 'Test1',
        'test1@gmail.com', '0000000',
        'Steven', 'King',
        'Sales',
        'Sales Manager'
    );
end;
```

```
/

--afisez angajatul nou
SELECT *
FROM emp
WHERE employee_id >= 1000;

-- Vezi câți subalterni direcți și indirecți are Steven King.
SELECT pachet_gest_companie.nr_subalterni('Steven', 'King')
FROM dual;

--afiseaza_angajati_per_job
BEGIN
    pachet_gest_companie.afiseaza_angajati_per_job;
END;
/
begin
pachet_gest_companie.gas_cel_mai_mic_comision(80,'Sales Manager');
end;
/
```

Worksheet | Query Builder

```
    SELECT * FROM emp
    WHERE job_id = id_job;
  -- Cursor care returnează toate joburile din companie.
    CURSOR toate_joburile
    RETURN jobs%ROWTYPE
    IS
    SELECT * FROM jobs;

    PROCEDURE afiseaza_angajati_per_job;
END;
/

CREATE OR REPLACE PACKAGE BODY pachet_companie
IS
    FUNCTION gaseste_job(
        nume jobs.job_title%TYPE
    ) RETURN jobs.job_id%TYPE
    IS
        v_id jobs.job_id%TYPE;
    BEGIN
```

Worksheet | Query Builder

```
-- Fac copii pentru tabelele date pentru a putea face modificari
DROP TABLE emp;
CREATE TABLE emp AS SELECT * FROM employees;
/
DROP TABLE dept;
CREATE TABLE dept AS SELECT * FROM departments;
DROP TABLE job_hist;
CREATE TABLE job_hist AS SELECT * FROM job_history;
DROP SEQUENCE emp_seq;
CREATE SEQUENCE emp_seq
START WITH 1000
INCREMENT BY 1
NOCACHE NOCYCLE;
/
COMMIT;
/

CREATE OR REPLACE PACKAGE pachet_gest_companie
IS
    -- gaseste jobul dat prin parametru
    FUNCTION gas_job(
        nume jobs.job_title%TYPE
    ) RETURN jobs.job_id%TYPE;
-- gaseste angajatul dat prin parametru
    FUNCTION gas_angajat(
        prenume emp.first_name%TYPE,
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.025 seconds

	PACHET_GEST_COMPANIE.NR_SUBALTERNI('STEVEN','KING')
1	107

```
BEGIN
    pachet_gest_companie.afiseaza_angajati_per_job;
```

Script Output | Query Result

Task completed in 0.21 seconds

```
Sundar Ande - 0
Amit Banda - 0
Lisa Ozer - 0
Harrison Bloom - 0
Tayler Fox - 0
William Smith - 0
Elizabeth Bates - 0
Sundita Kumar - 0
Ellen Abel - 0
Alyssa Hutton - 0
Jonathon Taylor - 1
Jack Livingston - 0
Kimberely Grant - 0
Charles Johnson - 0
Jobul Purchasing Manager:
Den Raphaely - 0
Jobul Purchasing Clerk:
Alexander Khoo - 0
```

```
--afisez angajatul nou
SELECT *
FROM emp
WHERE employee_id >= 1000;

-- Vezi câţi subalterni direcţi şi indirecţi are Steven King.
SELECT pachet_gest_companie.nr_subalterni('Steven', 'King')
FROM dual;

--afiseaza_angajati_per_job
BEGIN
    pachet_gest_companie.afiseaza_angajati_per_job;
END;
/
```

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.024 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	1000	John	John	john@john.com	0752411523	21-MAY-20	AD_VP	1000	(null)	(null)	(null)
2	1001	Ang1	Test1	test1@gmail.com	0000000	15-MAY-21	SA_MAN	11550	0	100	80

PLSQL6 EX1

```
-- 1
DROP TABLE dept;
CREATE TABLE dept AS (SELECT * FROM departments);
-- trigger la nivel de linie care sa-i permita doar utilizatorului SCOTT sa stearga informatii din tabela DEPT.
CREATE OR REPLACE TRIGGER tr_scott
BEFORE DELETE ON dept
FOR EACH ROW
BEGIN
   IF USER != 'SCOTT' THEN
      RAISE_APPLICATION_ERROR(-20000, 'Utilizatorul nu este Scott!!');
```

```
    END IF;
END;
/
-- primesc eroare daca incerc sa sterg inregistrari folosind userul meu
DELETE FROM dept;
/
```

```
-- 1
DROP TABLE dept;
CREATE TABLE dept AS (SELECT * FROM departments);
-- trigger la nivel de linie care sa-i permita doar utilizatorului SCOTT sa stearga informatii din tabela DEPT.
CREATE OR REPLACE TRIGGER tr_scott
BEFORE DELETE ON dept
FOR EACH ROW
BEGIN
    IF USER != 'SCOTT' THEN
        RAISE_APPLICATION_ERROR(-20000, 'Utilizatorul nu este Scott!!');
    END IF;
END;
/
-- primesc eroare daca incerc sa sterg inregistrari folosind userul meu
DELETE FROM dept;
/

---- 2

DROP TABLE emp;
CREATE TABLE emp AS (SELECT * FROM employees);
```

Script Output | Query Result

Task completed in 0.252 seconds

```
Table DEPT dropped.


Table DEPT created.


Trigger TR_SCOTT compiled


Error starting at line : 816 in command -
DELETE FROM dept
Error report -
ORA-20000: Utilizatorul nu este Scott!!
ORA-06512: at "GRUPA241.TR_SCOTT", line 3
ORA-04088: error during execution of trigger 'GRUPA241.TR_SCOTT'
```

PLSQL6 EX2

-- 2

```
DROP TABLE emp;
CREATE TABLE emp AS (SELECT * FROM employees);

CREATE OR REPLACE TRIGGER trig_marire
BEFORE UPDATE OF commission_pct ON emp
```

```
FOR EACH ROW
BEGIN
   -- Dacă comisionul depășește 50%--->nu permit actualizarea.
   IF :NEW.commission_pct > 0.5 THEN
      RAISE_APPLICATION_ERROR(-20000, 'Nu se poate mari comisionul cu mai mult de 50% din
salariu!!!');
   END IF;
END;
/

-- vreau sa modific cu un comision de 0,76
UPDATE emp
SET commission_pct = 0.76
WHERE employee_id = 100;
/
```

```
CREATE OR REPLACE TRIGGER trig_marire
BEFORE UPDATE OF commission_pct ON emp
FOR EACH ROW
BEGIN
    -- Dacă comisionul depășește 50%--->nu permit actualizarea.
    IF :NEW.commission_pct > 0.5 THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu se poate mari comisionul cu mai mult de 50% din salariu!!!');
    END IF;
END;
/

-- vreau sa modific cu un comision de 0,76
UPDATE emp
SET commission_pct = 0.76
WHERE employee_id = 100;
/


---- 3
-- (Re)creez tabelul de angajați
DROP TABLE info_emp;
CREATE TABLE info_emp AS (SELECT * FROM employees);
```

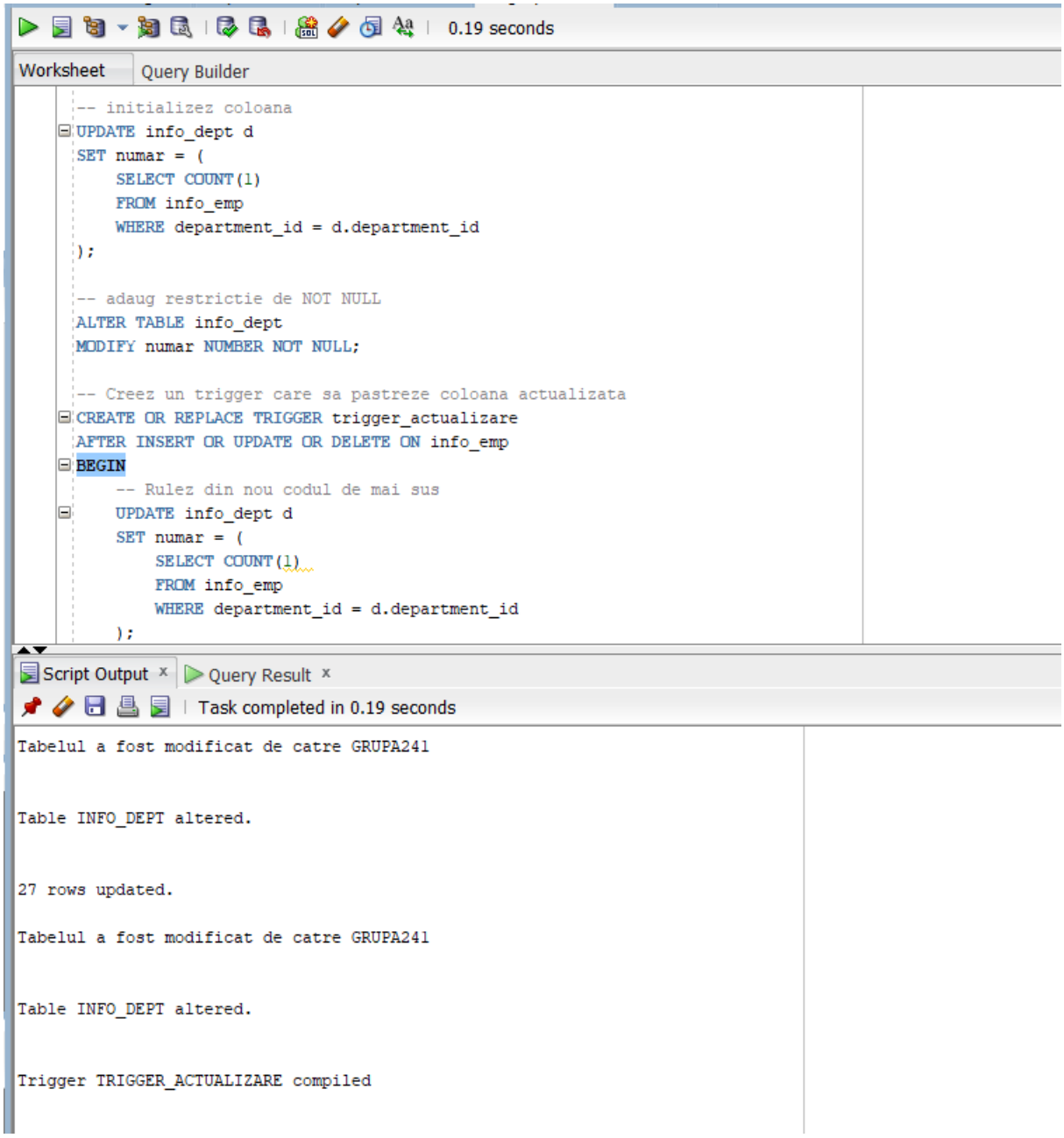
Script Output | Query Result

Task completed in 0.251 seconds

```
Table EMP created.


Trigger TRIG_MARIRE compiled


Error starting at line : 835 in command -
UPDATE emp
SET commission_pct = 0.76
WHERE employee_id = 100
Error report -
ORA-20000: Nu se poate mari comisionul cu mai mult de 50% din salariu!!!
ORA-06512: at "GRUPA241.TRIG_MARIRE", line 4
ORA-04088: error during execution of trigger 'GRUPA241.TRIG_MARIRE'
```

PLSQL6 EX3

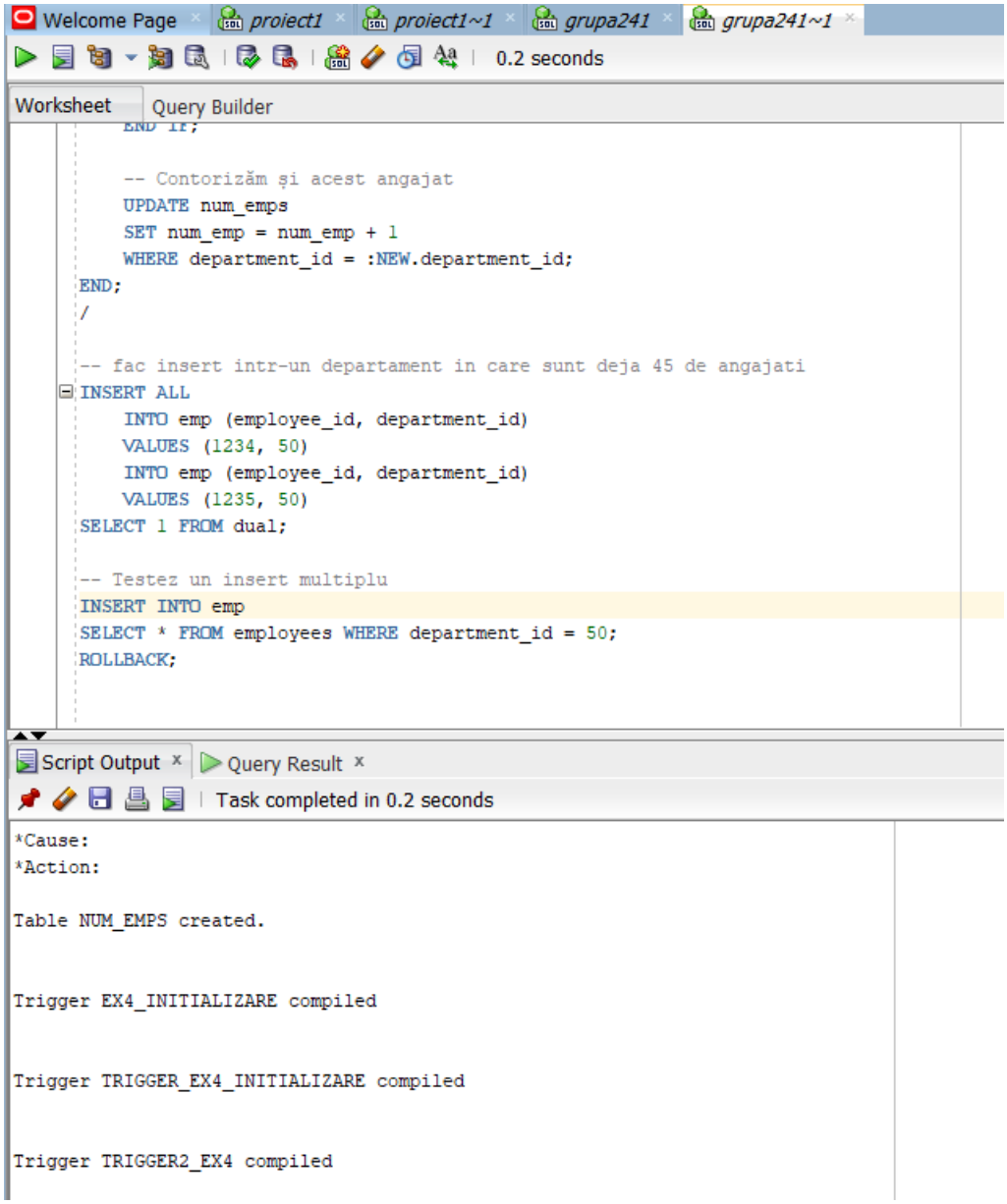
```
--3
DROP TABLE info_emp;
CREATE TABLE info_emp AS (SELECT * FROM employees);

DROP TABLE info_dept;
CREATE TABLE info_dept AS (SELECT * FROM departments);

--Adaug o noua coloana pt nr de angajati dintr-un departament
ALTER TABLE info_dept
ADD numar NUMBER;

-- initializez coloana
UPDATE info_dept d
SET numar = (
   SELECT COUNT(1)
   FROM info_emp
   WHERE department_id = d.department_id
);

-- adaug restrictie de NOT NULL
ALTER TABLE info_dept
MODIFY numar NUMBER NOT NULL;

-- Creez un trigger care sa pastreze coloana actualizata
CREATE OR REPLACE TRIGGER trigger_actualizare
AFTER INSERT OR UPDATE OR DELETE ON info_emp
BEGIN
   -- Rulez din nou codul de mai sus
   UPDATE info_dept d
   SET numar = (
      SELECT COUNT(1)
      FROM info_emp
      WHERE department_id = d.department_id
   );
END;
/
```



PLSQL6 EX4

```
-- 4
-- Creez o noua tabela in care stochez cati angajati am in fiecare departament
DROP TABLE nr_emps;
CREATE TABLE nr_emps AS (
   SELECT department_id, COUNT(1) AS num_emp
   FROM emp
   GROUP BY department_id
);
```

```
CREATE OR REPLACE TRIGGER trigger_initializare1
--triggerul inițializează tabela creata la fiecare insert/update
BEFORE INSERT OR UPDATE ON employees
BEGIN
   DELETE FROM num_emps;
 -- Inserez valori
   INSERT INTO nr_emps (department_id, num_emp)
   SELECT department_id, COUNT(1)
   FROM emp
   GROUP BY department_id;
END;
/
--trigger care sa se declanseze la insert-ul unui angajat intr-un departament in care lucreaza deja 45 de
persoane
CREATE OR REPLACE TRIGGER trigger2_ex4
BEFORE INSERT OR UPDATE ON emp
FOR EACH ROW
DECLARE
--declar o variabila in care incarc numarul angajatilor
   v_num_ang NUMBER;
BEGIN
   -- selectez nr de angajati
   SELECT num_emp INTO v_num_ang
   FROM num_emps
   WHERE department_id = :NEW.department_id;
   IF v_num_ang >= 45 THEN
--tratez cazul in care sunt mai mult de 45 de angajati
      RAISE_APPLICATION_ERROR(-20000, 'Prea multi angajati');
   END IF;
   -- daca sunt mai putin de 45 de angajati → facem update si contorizam si noul angajat
   UPDATE num_emps
   SET num_emp = num_emp + 1
   WHERE department_id = :NEW.department_id;
END;
/

-- fac insert intr-un departament in care sunt deja 45 de angajati
INSERT ALL
   INTO emp (employee_id, department_id)
   VALUES (1234, 50)
   INTO emp (employee_id, department_id)
   VALUES (1235, 50)
SELECT 1 FROM dual;
```



PLSQL6 EX5

```
-- 5
DROP TABLE emp_test;

CREATE TABLE emp_test AS (
   SELECT employee_id, last_name, first_name, department_id
   FROM employees
);

ALTER TABLE emp_test
ADD CONSTRAINT emp_test_pk PRIMARY KEY (employee_id);

CREATE TABLE dept_test AS (
```

```
    SELECT department_id, department_name
    FROM departments
);

ALTER TABLE dept_test
ADD CONSTRAINT dept_test_pk PRIMARY KEY (department_id);
--trigger stergere--
CREATE OR REPLACE TRIGGER trigg_stergere
AFTER DELETE ON dept_test
--trigger la nivel de rand--
FOR EACH ROW
BEGIN
    -- Șterg angajații care erau in acest departament
    DELETE FROM emp_test
    WHERE department_id = :OLD.department_id;
END;
/

CREATE OR REPLACE TRIGGER trigg_modificare
AFTER UPDATE ON dept_test
FOR EACH ROW
BEGIN
    -- Modific angajații care erau deja în departament.
    UPDATE emp_test
    SET department_id = :NEW.department_id
    WHERE department_id = :OLD.department_id;
END;
/

--testare
ALTER TABLE emp_test
DROP CONSTRAINT emp_test_dept_fk;

ALTER TABLE emp_test
ADD CONSTRAINT emp_test_dept_fk
    FOREIGN KEY (department_id)
    REFERENCES dept_test(department_id);

ALTER TABLE emp_test
ADD CONSTRAINT emp_test_dept_fk
    FOREIGN KEY (department_id)
    REFERENCES dept_test(department_id)
    ON DELETE CASCADE;

ALTER TABLE emp_test
ADD CONSTRAINT emp_test_dept_fk
    FOREIGN KEY (department_id)
    REFERENCES dept_test(department_id)
    ON DELETE SET NULL;
```

0.16599999 seconds

Worksheet | Query Builder

```
/

--testare
ALTER TABLE emp_test
DROP CONSTRAINT emp_test_dept_fk;

ALTER TABLE emp_test
ADD CONSTRAINT emp_test_dept_fk
    FOREIGN KEY (department_id)
    REFERENCES dept_test(department_id);

ALTER TABLE emp_test
ADD CONSTRAINT emp_test_dept_fk
    FOREIGN KEY (department_id)
    REFERENCES dept_test(department_id)
```

Script Output | Query Result

Task completed in 0.166 seconds

```
Error report -
ORA-02443: Cannot drop constraint  - nonexistent constraint
02443. 00000 -  "Cannot drop constraint  - nonexistent constraint"
*Cause:    alter table drop constraint <constraint_name>
*Action:   make sure you supply correct constraint name.
Tabelul a fost modificat de catre GRUPA241


Table EMP_TEST altered.

Tabelul a fost modificat de catre GRUPA241


Error starting at line : 998 in command -
ALTER TABLE emp_test
ADD CONSTRAINT emp_test_dept_fk
    FOREIGN KEY (department_id)
    REFERENCES dept_test(department_id)
    ON DELETE CASCADE
Error report -
ORA-02275: such a referential constraint already exists in the table
02275. 00000 -  "such a referential constraint already exists in the table"
*Cause:    Self-evident.
*Action:   Remove the extra constraint.
```

PLSQL6 EX 6

```
-- 6

DROP TABLE database_errors;
CREATE TABLE database_errors (
   user_id NVARCHAR2(100),
   nume_bd NVARCHAR2(100),
   erori NVARCHAR2(2000),
   data DATE
);

--trigger insert
CREATE OR REPLACE TRIGGER trigger_insert
AFTER SERVERERROR
ON DATABASE
BEGIN
```

```
    INSERT INTO database_errors
    VALUES (
        SYS.LOGIN_USER,
        SYS.DATABASE_NAME,
        DBMS.FORMAT_ERROR_STACK,
        SYSDATE
    );
END;
```

Nu am suficiente permisiuni.

Worksheet | Query Builder

```
-- 6

DROP TABLE database_errors;
CREATE TABLE database_errors (
    user_id NVARCHAR2(100),
    nume_bd NVARCHAR2(100),
    erori NVARCHAR2(2000),
    data DATE
);

--trigger insert
CREATE OR REPLACE TRIGGER trigger_insert
AFTER SERVERERROR
ON DATABASE
BEGIN
    INSERT INTO database_errors
    VALUES (
        SYS.LOGIN_USER,
        SYS.DATABASE_NAME,
        DBMS.FORMAT_ERROR_STACK,
        SYSDATE
    );
END;


---- Extra
-- Să se creeze un trigger check_sal_*** care garantează ca, ori de câte ori un angajat nou este
```

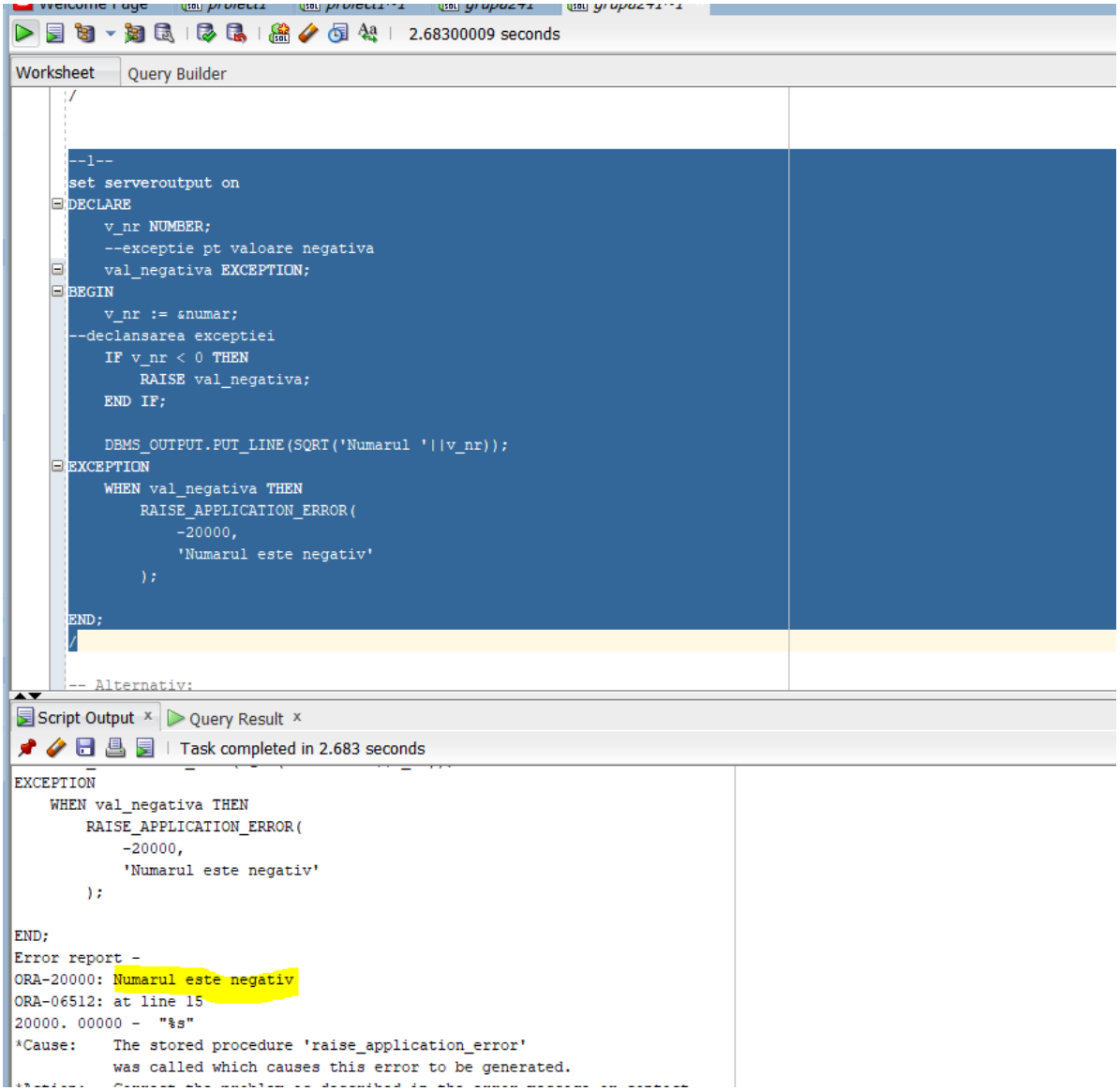
Script Output | Query Result

Task completed in 0.15 seconds

```
        SYS.DATABASE_NAME,
        DBMS.FORMAT_ERROR_STACK,
        SYSDATE
    );
END;
Error report -
ORA-01031: insufficient privileges
01031. 00000 -  "insufficient privileges"
*Cause:    An attempt was made to perform a database operation without
           the necessary privileges.
*Action:   Ask your database administrator or designated security
           administrator to grant you the necessary privileges
```

PLSQL7 EX1

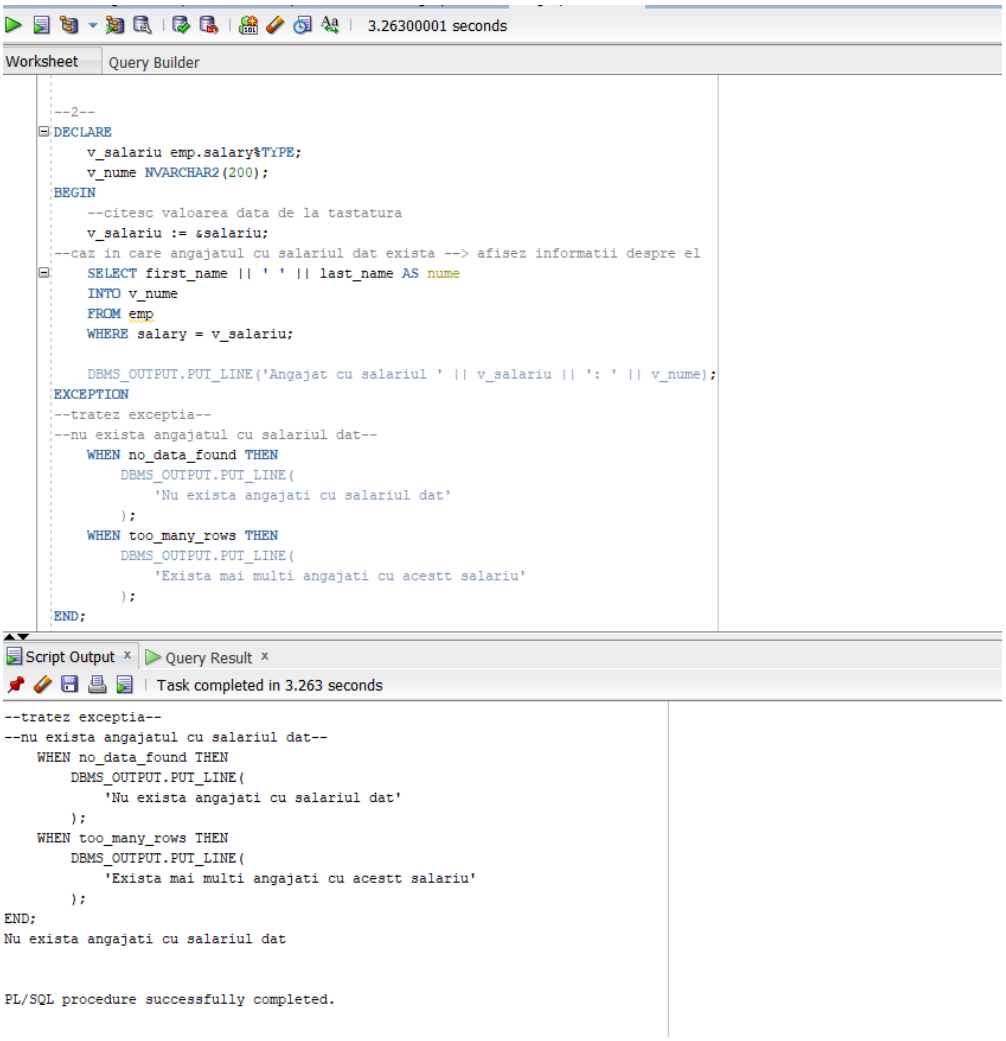
```
--1--
set serveroutput on
DECLARE
   v_nr NUMBER;
   --exceptie pt valoare negativa
   val_negativa EXCEPTION;
BEGIN
   v_nr := &numar;
--declansarea exceptiei
   IF v_nr < 0 THEN
      RAISE val_negativa;
   END IF;

   DBMS_OUTPUT.PUT_LINE(SQRT('Numarul '||v_nr));
EXCEPTION
   WHEN val_negativa THEN
      RAISE_APPLICATION_ERROR(
         -20000,
         'Numarul este negativ'
      );

END;
/
```



PLSQL7 EX2

```
--2--
DECLARE
   v_salariu emp.salary%TYPE;
   v_nume NVARCHAR2(200);
BEGIN
   --citesc valoarea data de la tastatura
   v_salariu := &salariu;
--caz in care angajatul cu salariul dat exista --> afisez informatii despre el
   SELECT first_name || ' ' || last_name AS nume
   INTO v_nume
   FROM emp
   WHERE salary = v_salariu;
```

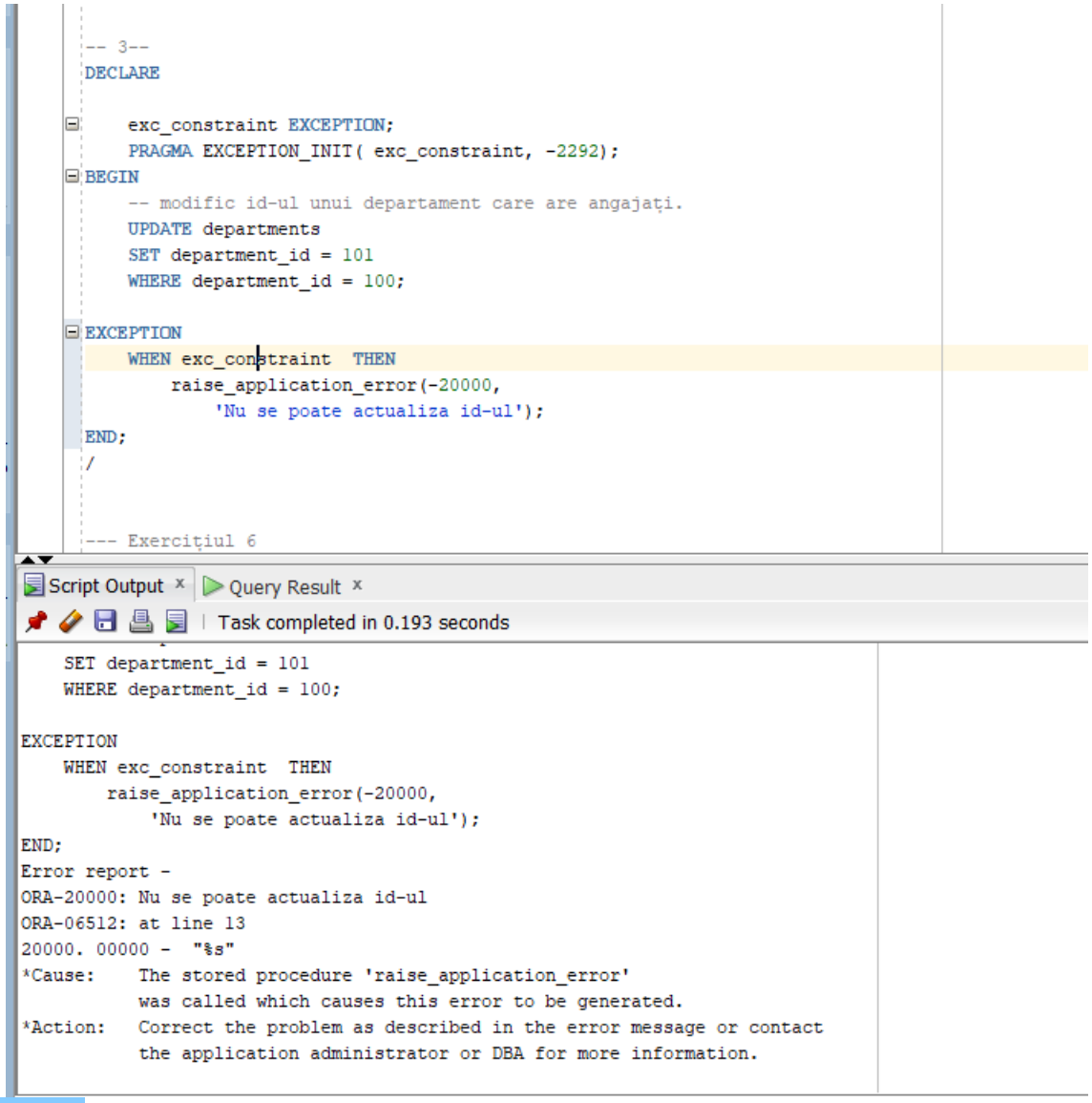
```
   DBMS_OUTPUT.PUT_LINE('Angajat cu salariul ' || v_salariu || ': ' || v_nume);
EXCEPTION
--tratez exceptia--
--nu exista angajatul cu salariul dat--
   WHEN no_data_found THEN
      DBMS_OUTPUT.PUT_LINE(
         'Nu exista angajati cu salariul dat'
      );
   WHEN too_many_rows THEN
      DBMS_OUTPUT.PUT_LINE(
         'Exista mai multi angajati cu acestt salariu'
      );
END;
/
```



PLSQL7 EX3

```
-- 3--
DECLARE

   exc_constraint EXCEPTION;
```

```
   PRAGMA EXCEPTION_INIT( exc_constraint, -2292);
BEGIN
   -- modific id-ul unui departament care are angajați.
   UPDATE departments
   SET department_id = 101
   WHERE department_id = 100;

EXCEPTION
   WHEN exc_constraint  THEN
      raise_application_error(-20000,
         'Nu se poate actualiza id-ul');
END;
/
```



PLSQL7 EX6

-- 6--
set serveroutput on

```
DECLARE
   v_nume departments.department_name%TYPE;
BEGIN
   BEGIN
   --incarc numele departamentului in variabila v_nume
   --fac join pt a gasi departamenul dintr-o anumita locatie
      SELECT department_name INTO v_nume
      FROM departments
      INNER JOIN locations
      USING (location_id)
      WHERE city LIKE &oras;

      DBMS_OUTPUT.PUT_LINE('Nume departament--> ' || v_nume);
   EXCEPTION
      WHEN no_data_found THEN
      --tratez cazul in care nu am gasit departamenul cu locatia data
         raise_application_error(
            -20000,
            'Nu am gasit un departamentul!!'
         );
   END;

   BEGIN

      SELECT department_name INTO v_nume
      FROM departments
      --introduc id-ul departamentului de la tastatura
      WHERE department_id = &id_departament;

      DBMS_OUTPUT.PUT_LINE('Nume departament: ' || v_nume);
   EXCEPTION
      WHEN no_data_found THEN
         raise_application_error(
            -20001,
            'Nu am gasit un departament cu id-ul dat'
         );
   END;
END;
/
```