# Procedura stocată e4\_oras\_angajat (modificare simplă a funcției din tema s8)

|  |
| --- |
| /\*      E7. Adaptați cerințele exercițiilor 2 și 4 (folosind ca bază cerințele exercițiilor 1, respectiv 3) pentru      diagrama proiectului prezentată la materia Baze de Date din anul I. Rezolvați aceste două exerciții      în PL/SQL, folosind baza de date proprie. (PARTIAL doar ex. 4)      1. (adaptat)      Definiți un subprogram prin care să obțineți orașul în care lucrează un angajat, folosind numele angajatului.      Funcția trebuie să funcționeze chiar dacă utilizatorul introduce doar numele de familie sau doar prenumele.      Tratați toate excepțiile ce pot fi generate.      4. Rezolvați exercițiul 1 folosind o procedură stocată.  \*/  CREATE OR REPLACE PROCEDURE e4\_oras\_angajat(v\_nume IN angajat.nume%TYPE)  IS      v\_id angajat.id\_angajat%TYPE;      v\_nume\_oras oras.nume%TYPE;      BEGIN          SELECT id\_angajat          INTO v\_id          FROM angajat          WHERE regexp\_like(nume, '(^| )'||v\_nume||'($| )', 'i');          -- Imposibil să avem expectie la join, datorită constrângerilor          -- FOREIGN KEY și NOT NULL pe id\_restaurant și id\_oras.          SELECT oras.nume          INTO v\_nume\_oras          FROM angajat          JOIN restaurant          USING (id\_restaurant)          JOIN oras          USING (id\_oras)          WHERE id\_angajat = v\_id;          dbms\_output.PUT\_LINE('Lucrează în '||v\_nume\_oras||'.');      EXCEPTION          WHEN NO\_DATA\_FOUND THEN              RAISE\_APPLICATION\_ERROR(-20009, 'Nu s-a găsit un angajat cu nume sau prenume '||v\_nume||'.');          WHEN TOO\_MANY\_ROWS THEN              RAISE\_APPLICATION\_ERROR(-20009, 'Input ambiguu: multiplii angajați găsiți cu nume sau prenume '||v\_nume||'.');          WHEN OTHERS THEN              RAISE\_APPLICATION\_ERROR(-20009,'Alta eroare: '||SQLERRM);  END;  / |

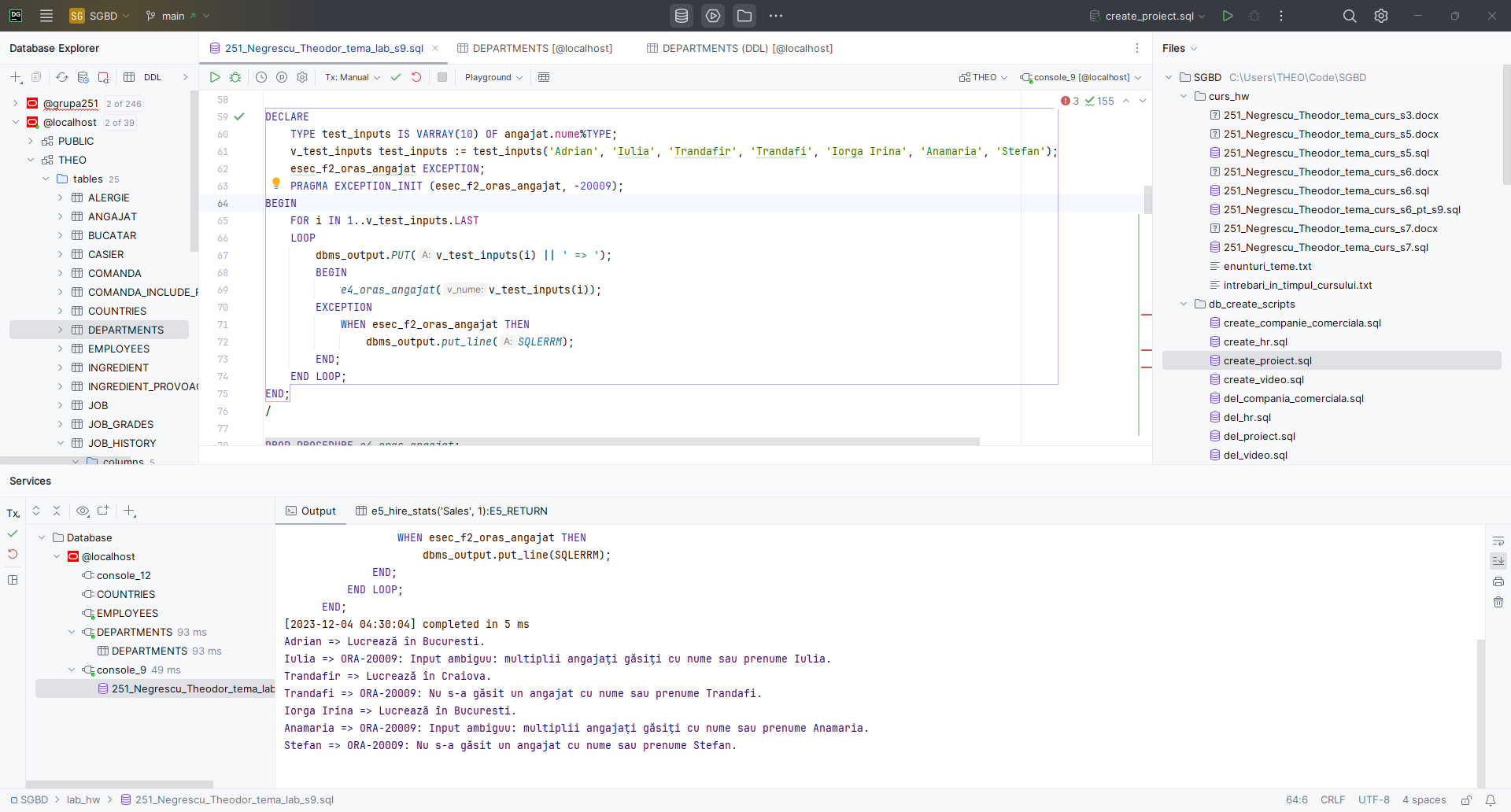
# Găsirea inputurilor ambigue pentru testarea codului (nicio schimbare de la tema s8)

|  |
| --- |
| /\*      Găsește un nume a.î. count >= 1 pentru a cauza eroarea  \*/  WITH fn AS (      SELECT regexp\_substr(nume, '(\w+)$', 1, 1, NULL, 1) AS nume FROM angajat  )  SELECT nume, COUNT(\*) FROM fn  GROUP BY nume  ORDER BY COUNT(\*) DESC;  /\*   Iulia si Anamaria ar trebui să cauzeze eroarea.  \*/ |

# Codul PL/SQL pentru a testa funcția stocată (modificare simplă din tema s8)

|  |
| --- |
| DECLARE      TYPE test\_inputs IS VARRAY(10) OF angajat.nume%TYPE;      v\_test\_inputs test\_inputs := test\_inputs('Adrian', 'Iulia', 'Trandafir', 'Trandafi', 'Iorga Irina', 'Anamaria', 'Stefan');      esec\_f2\_oras\_angajat EXCEPTION;      PRAGMA EXCEPTION\_INIT (esec\_f2\_oras\_angajat, -20009);  BEGIN      FOR i IN 1..v\_test\_inputs.LAST      LOOP          dbms\_output.PUT(v\_test\_inputs(i) || ' => ');          BEGIN              e4\_oras\_angajat(v\_test\_inputs(i));          EXCEPTION              WHEN esec\_f2\_oras\_angajat THEN                  dbms\_output.put\_line(SQLERRM);          END;      END LOOP;  END;  / |

# Screenshot e4\_oras\_angajat



Deoarece procedura de la exercițiul 4 a fost simplu de creat prin modificarea codului de la funcția stocată de la tema anterioară (am mutat printarea numelui orașului din blocul anonim în procedură), am rezolvat și E5+E6 pentru a adăuga niște complexitate la tema asta.

# Funcția stocată e5\_hire\_stats

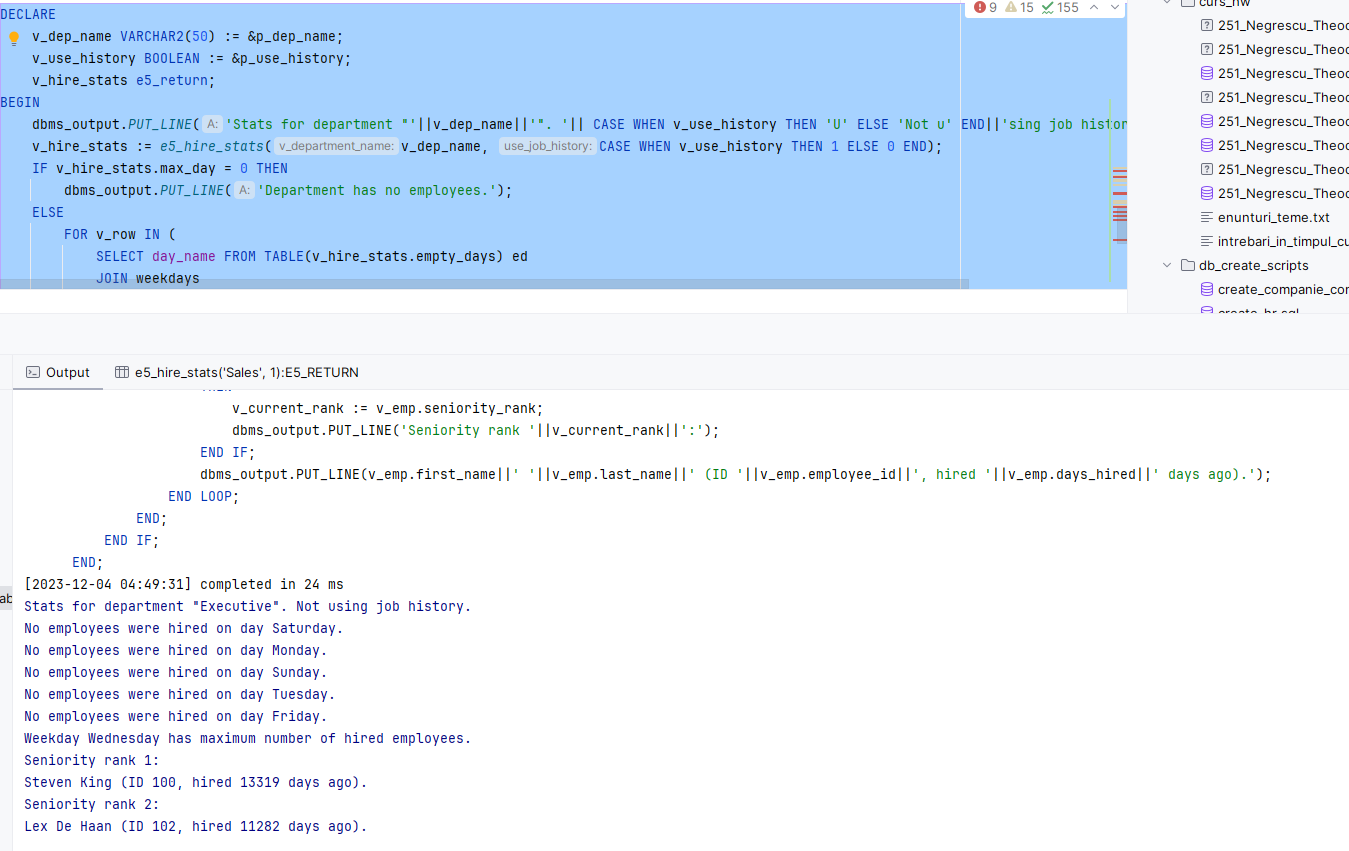
|  |
| --- |
| /\*  E5. Definiți un subprogram care obține pentru fiecare nume de departament ziua din săptămână în  care au fost angajate cele mai multe persoane, lista cu numele acestora, vechimea și venitul lor  lunar. Afișați mesaje corespunzătoare următoarelor cazuri:  - într-un departament nu lucrează niciun angajat;  - într-o zi din săptămână nu a fost nimeni angajat.  Observații:  a.  Numele departamentului și ziua apar o singură dată în rezultat.  b.  Rezolvați problema în două variante, după cum se ține  cont  sau  nu  de  istoricul  joburilor  angajaților.  E6. Modificați exercițiul anterior astfel încât lista cu numele angajaților să apară într-un clasament  creat în funcție de vechimea acestora în departament. Specificați  numărul  poziției din  clasament  și  apoi  lista  angajaților  care  ocupă acel loc. Dacă doi angajați  au  aceeași  vechime,  atunci aceștia ocupă aceeași poziție în clasament.  \*/  CREATE TYPE e5\_employee IS OBJECT (employee\_id NUMBER(6), first\_name VARCHAR2(50), last\_name VARCHAR2(50), days\_hired NUMBER(10), seniority\_rank NUMBER(6));  CREATE TYPE e5\_employees IS TABLE OF e5\_employee;  CREATE TYPE e5\_empty\_days IS TABLE OF NUMBER(6);  CREATE TYPE e5\_return IS OBJECT (empty\_days e5\_empty\_days, max\_day NUMBER(1), max\_day\_employees e5\_employees);  CREATE TYPE emp\_hire\_date IS OBJECT (employee\_id NUMBER(6), hire\_date DATE);  CREATE TYPE emp\_hire\_dates IS TABLE OF emp\_hire\_date;  CREATE TYPE emp\_hire\_day IS OBJECT (employee\_id NUMBER(6), hire\_day NUMBER(1));  CREATE TYPE emp\_hire\_days IS TABLE OF emp\_hire\_day;  CREATE TABLE weekdays AS  SELECT 1 day\_nr, 'Monday' day\_name FROM DUAL  UNION  SELECT 2, 'Tuesday' FROM DUAL  UNION  SELECT 3, 'Wednesday' FROM DUAL  UNION  SELECT 4, 'Thursday' FROM DUAL  UNION  SELECT 5, 'Friday' FROM DUAL  UNION  SELECT 6, 'Saturday' FROM DUAL  UNION  SELECT 7, 'Sunday' FROM DUAL;  CREATE OR REPLACE FUNCTION e5\_hire\_stats(v\_department\_name VARCHAR2, use\_job\_history NUMBER)  RETURN e5\_return  IS      v\_retval e5\_return := e5\_return(e5\_empty\_days(), 0, e5\_employees());      v\_dep\_id NUMBER(4);      v\_hire\_dates emp\_hire\_dates;      v\_hire\_days emp\_hire\_days;      BEGIN          SELECT department\_id          INTO v\_dep\_id          FROM departments          WHERE department\_name = v\_department\_name;          SELECT emp\_hire\_date(employee\_id, hire\_date)          BULK COLLECT INTO v\_hire\_dates          FROM employees          WHERE department\_id = v\_dep\_id;          IF use\_job\_history != 0          THEN              DECLARE                  tmp emp\_hire\_dates := v\_hire\_dates;              BEGIN                  WITH hire\_dates AS (                      SELECT employee\_id, hire\_date                      FROM TABLE(tmp)                      UNION                      SELECT employee\_id, jh.end\_date AS hire\_date                      FROM TABLE(tmp)                      JOIN job\_history jh                      USING (employee\_id)                  )                  SELECT emp\_hire\_date(t1.employee\_id, t1.hire\_date)                  BULK COLLECT INTO v\_hire\_dates                  FROM hire\_dates t1                  LEFT JOIN hire\_dates t2                  ON t1.employee\_id = t2.employee\_id AND t1.hire\_date < t2.hire\_date                  WHERE t2.hire\_date IS NULL;              END;          END IF;          SELECT emp\_hire\_day(employee\_id, TRUNC(hire\_date) - TRUNC(hire\_date, 'IW') + 1) AS hire\_day          BULK COLLECT INTO v\_hire\_days          FROM TABLE(v\_hire\_dates);          SELECT day\_nr          BULK COLLECT INTO v\_retval.empty\_days          FROM weekdays          LEFT JOIN TABLE(v\_hire\_days) hire\_days          ON weekdays.day\_nr = hire\_days.hire\_day          WHERE hire\_days.hire\_day IS NULL;          IF v\_retval.empty\_days.LAST = 7 THEN              RETURN v\_retval;          END IF;          WITH day\_counts AS (              SELECT hire\_day              FROM TABLE(v\_hire\_days)              GROUP BY hire\_day              ORDER BY COUNT(\*) DESC          )          SELECT hire\_day          INTO v\_retval.max\_day          FROM day\_counts          WHERE rownum = 1;          WITH emp\_to\_rank AS (              SELECT employee\_id, first\_name, last\_name, SYSDATE - hire\_date AS days\_worked              FROM employees              JOIN TABLE (v\_hire\_days)              USING (employee\_id)              WHERE hire\_day = v\_retval.max\_day              ORDER BY days\_worked DESC          )          SELECT e5\_employee(employee\_id, first\_name, last\_name, days\_worked, RANK() OVER (ORDER BY days\_worked DESC))          BULK COLLECT INTO v\_retval.max\_day\_employees          FROM emp\_to\_rank;          RETURN v\_retval;  END;  / |

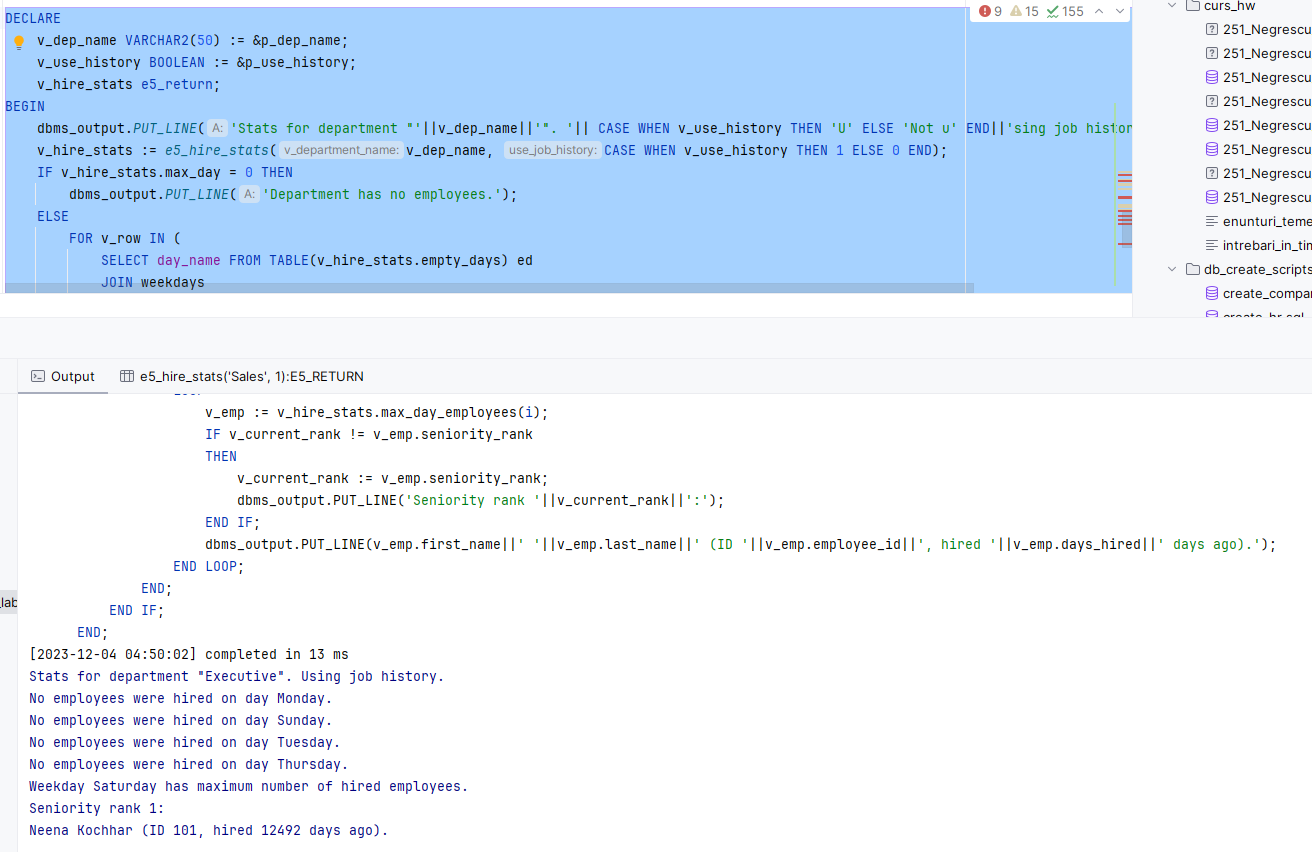
# Blocul anonim pentru utilizarea funcției e5\_hire\_stats

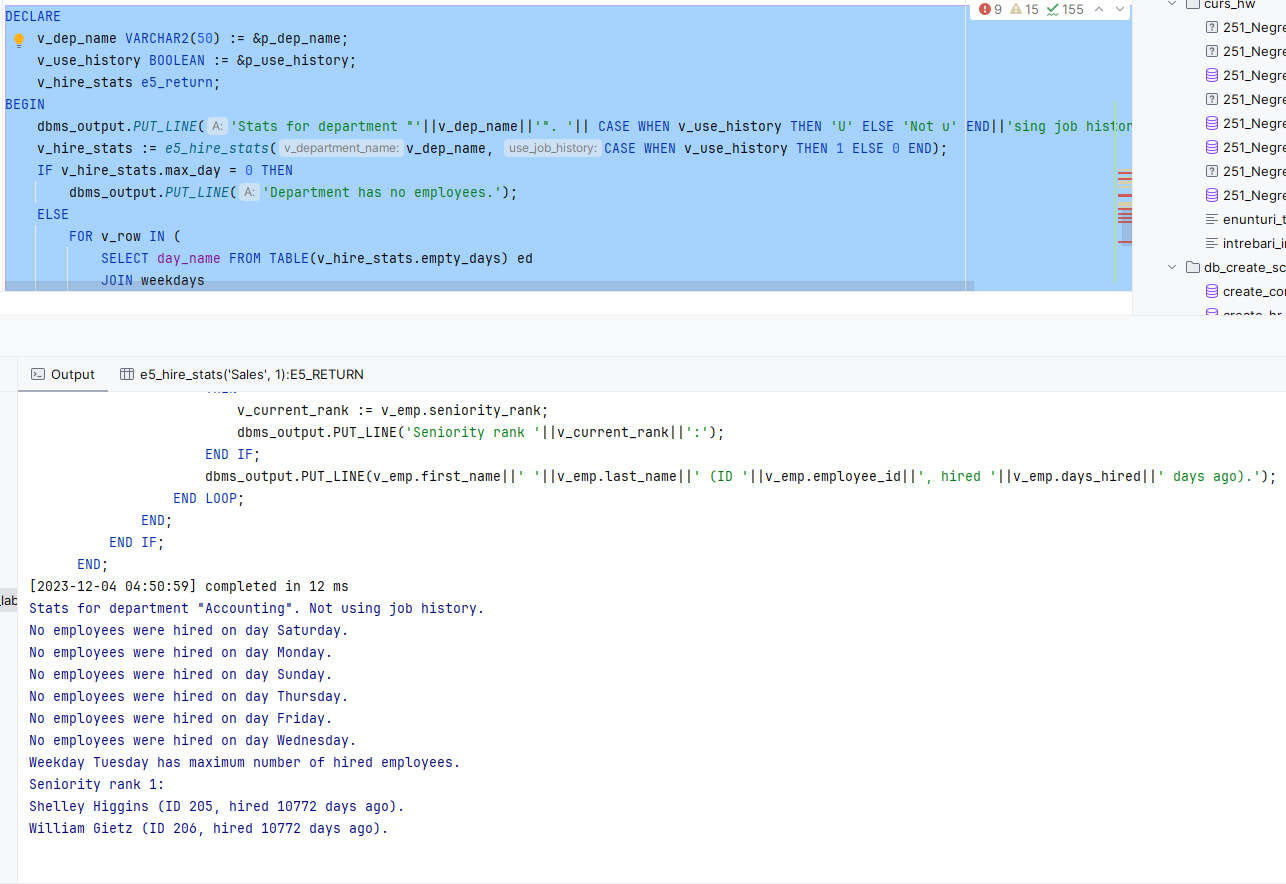
|  |
| --- |
| DECLARE      v\_dep\_name VARCHAR2(50) := &p\_dep\_name;      v\_use\_history BOOLEAN := &p\_use\_history;      v\_hire\_stats e5\_return;  BEGIN      dbms\_output.PUT\_LINE('Stats for department "'||v\_dep\_name||'". '|| CASE WHEN v\_use\_history THEN 'U' ELSE 'Not u' END||'sing job history.');      v\_hire\_stats := e5\_hire\_stats(v\_dep\_name, CASE WHEN v\_use\_history THEN 1 ELSE 0 END);      IF v\_hire\_stats.max\_day = 0 THEN          dbms\_output.PUT\_LINE('Department has no employees.');      ELSE          FOR v\_row IN (              SELECT day\_name FROM TABLE(v\_hire\_stats.empty\_days) ed              JOIN weekdays              ON weekdays.day\_nr = ed.column\_value          )          LOOP              dbms\_output.PUT\_LINE('No employees were hired on day '||v\_row.day\_name||'.');          END LOOP;          DECLARE              v\_day\_name VARCHAR2(50);          BEGIN              SELECT day\_name              INTO v\_day\_name              FROM weekdays              WHERE day\_nr = v\_hire\_stats.max\_day;              dbms\_output.PUT\_LINE('Weekday '||v\_day\_name||' has maximum number of hired employees.');          END;          DECLARE              v\_current\_rank BINARY\_INTEGER := 0;              v\_emp e5\_employee;          BEGIN              FOR i IN 1..v\_hire\_stats.max\_day\_employees.last              LOOP                  v\_emp := v\_hire\_stats.max\_day\_employees(i);                  IF v\_current\_rank != v\_emp.seniority\_rank                  THEN                      v\_current\_rank := v\_emp.seniority\_rank;                      dbms\_output.PUT\_LINE('Seniority rank '||v\_current\_rank||':');                  END IF;                  dbms\_output.PUT\_LINE(v\_emp.first\_name||' '||v\_emp.last\_name||' (ID '||v\_emp.employee\_id||', hired '||v\_emp.days\_hired||' days ago).');              END LOOP;          END;      END IF;  END; |

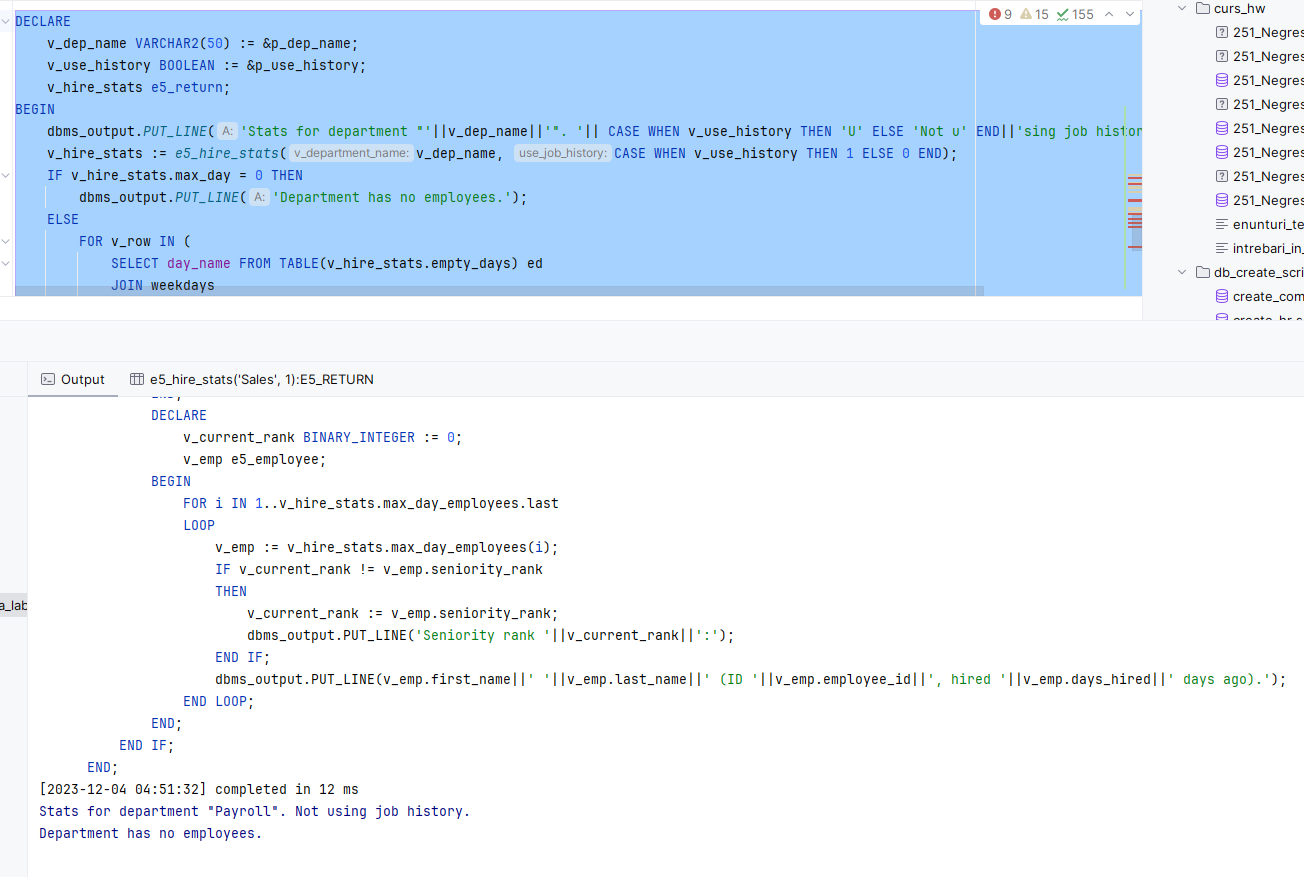
# Screenshot-uri e5\_hire\_stats

Screenshot-urile afișează erori deoarece DataGrip nu înțelege folosirea de WITH în PL/SQL, și pare să aibă probleme cu tipuri OBJECT.









# Întregul fișier 251\_Negrescu\_Theodor\_tema\_lab\_s9.sql

|  |
| --- |
| /\*      E7. Adaptați cerințele exercițiilor 2 și 4 (folosind ca bază cerințele exercițiilor 1, respectiv 3) pentru      diagrama proiectului prezentată la materia Baze de Date din anul I. Rezolvați aceste două exerciții      în PL/SQL, folosind baza de date proprie. (PARTIAL doar ex. 4)      1. (adaptat)      Definiți un subprogram prin care să obțineți orașul în care lucrează un angajat, folosind numele angajatului.      Funcția trebuie să funcționeze chiar dacă utilizatorul introduce doar numele de familie sau doar prenumele.      Tratați toate excepțiile ce pot fi generate.      4. Rezolvați exercițiul 1 folosind o procedură stocată.  \*/  CREATE OR REPLACE PROCEDURE e4\_oras\_angajat(v\_nume IN angajat.nume%TYPE)  IS      v\_id angajat.id\_angajat%TYPE;      v\_nume\_oras oras.nume%TYPE;      BEGIN          SELECT id\_angajat          INTO v\_id          FROM angajat          WHERE regexp\_like(nume, '(^| )'||v\_nume||'($| )', 'i');          -- Imposibil să avem expectie la join, datorită constrângerilor          -- FOREIGN KEY și NOT NULL pe id\_restaurant și id\_oras.          SELECT oras.nume          INTO v\_nume\_oras          FROM angajat          JOIN restaurant          USING (id\_restaurant)          JOIN oras          USING (id\_oras)          WHERE id\_angajat = v\_id;          dbms\_output.PUT\_LINE('Lucrează în '||v\_nume\_oras||'.');      EXCEPTION          WHEN NO\_DATA\_FOUND THEN              RAISE\_APPLICATION\_ERROR(-20009, 'Nu s-a găsit un angajat cu nume sau prenume '||v\_nume||'.');          WHEN TOO\_MANY\_ROWS THEN              RAISE\_APPLICATION\_ERROR(-20009, 'Input ambiguu: multiplii angajați găsiți cu nume sau prenume '||v\_nume||'.');          WHEN OTHERS THEN              RAISE\_APPLICATION\_ERROR(-20009,'Alta eroare: '||SQLERRM);  END;  /  /\*      Găsește un nume a.î. count >= 1 pentru a cauza eroarea  \*/  WITH fn AS (      SELECT regexp\_substr(nume, '(\w+)$', 1, 1, NULL, 1) AS nume FROM angajat  )  SELECT nume, COUNT(\*) FROM fn  GROUP BY nume  ORDER BY COUNT(\*) DESC;  /\*   Iulia si Anamaria ar trebui să cauzeze eroarea.  \*/  DECLARE      TYPE test\_inputs IS VARRAY(10) OF angajat.nume%TYPE;      v\_test\_inputs test\_inputs := test\_inputs('Adrian', 'Iulia', 'Trandafir', 'Trandafi', 'Iorga Irina', 'Anamaria', 'Stefan');      esec\_f2\_oras\_angajat EXCEPTION;      PRAGMA EXCEPTION\_INIT (esec\_f2\_oras\_angajat, -20009);  BEGIN      FOR i IN 1..v\_test\_inputs.LAST      LOOP          dbms\_output.PUT(v\_test\_inputs(i) || ' => ');          BEGIN              e4\_oras\_angajat(v\_test\_inputs(i));          EXCEPTION              WHEN esec\_f2\_oras\_angajat THEN                  dbms\_output.put\_line(SQLERRM);          END;      END LOOP;  END;  /  DROP PROCEDURE e4\_oras\_angajat;  /\*  E5. Definiți un subprogram care obține pentru fiecare nume de departament ziua din săptămână în  care au fost angajate cele mai multe persoane, lista cu numele acestora, vechimea și venitul lor  lunar. Afișați mesaje corespunzătoare următoarelor cazuri:  - într-un departament nu lucrează niciun angajat;  - într-o zi din săptămână nu a fost nimeni angajat.  Observații:  a.  Numele departamentului și ziua apar o singură dată în rezultat.  b.  Rezolvați problema în două variante, după cum se ține  cont  sau  nu  de  istoricul  joburilor  angajaților.  E6. Modificați exercițiul anterior astfel încât lista cu numele angajaților să apară într-un clasament  creat în funcție de vechimea acestora în departament. Specificați  numărul  poziției din  clasament  și  apoi  lista  angajaților  care  ocupă acel loc. Dacă doi angajați  au  aceeași  vechime,  atunci aceștia ocupă aceeași poziție în clasament.  \*/  CREATE TYPE e5\_employee IS OBJECT (employee\_id NUMBER(6), first\_name VARCHAR2(50), last\_name VARCHAR2(50), days\_hired NUMBER(10), seniority\_rank NUMBER(6));  CREATE TYPE e5\_employees IS TABLE OF e5\_employee;  CREATE TYPE e5\_empty\_days IS TABLE OF NUMBER(6);  CREATE TYPE e5\_return IS OBJECT (empty\_days e5\_empty\_days, max\_day NUMBER(1), max\_day\_employees e5\_employees);  CREATE TYPE emp\_hire\_date IS OBJECT (employee\_id NUMBER(6), hire\_date DATE);  CREATE TYPE emp\_hire\_dates IS TABLE OF emp\_hire\_date;  CREATE TYPE emp\_hire\_day IS OBJECT (employee\_id NUMBER(6), hire\_day NUMBER(1));  CREATE TYPE emp\_hire\_days IS TABLE OF emp\_hire\_day;  CREATE TABLE weekdays AS  SELECT 1 day\_nr, 'Monday' day\_name FROM DUAL  UNION  SELECT 2, 'Tuesday' FROM DUAL  UNION  SELECT 3, 'Wednesday' FROM DUAL  UNION  SELECT 4, 'Thursday' FROM DUAL  UNION  SELECT 5, 'Friday' FROM DUAL  UNION  SELECT 6, 'Saturday' FROM DUAL  UNION  SELECT 7, 'Sunday' FROM DUAL;  CREATE OR REPLACE FUNCTION e5\_hire\_stats(v\_department\_name VARCHAR2, use\_job\_history NUMBER)  RETURN e5\_return  IS      v\_retval e5\_return := e5\_return(e5\_empty\_days(), 0, e5\_employees());      v\_dep\_id NUMBER(4);      v\_hire\_dates emp\_hire\_dates;      v\_hire\_days emp\_hire\_days;      BEGIN          SELECT department\_id          INTO v\_dep\_id          FROM departments          WHERE department\_name = v\_department\_name;          SELECT emp\_hire\_date(employee\_id, hire\_date)          BULK COLLECT INTO v\_hire\_dates          FROM employees          WHERE department\_id = v\_dep\_id;          IF use\_job\_history != 0          THEN              DECLARE                  tmp emp\_hire\_dates := v\_hire\_dates;              BEGIN                  WITH hire\_dates AS (                      SELECT employee\_id, hire\_date                      FROM TABLE(tmp)                      UNION                      SELECT employee\_id, jh.end\_date AS hire\_date                      FROM TABLE(tmp)                      JOIN job\_history jh                      USING (employee\_id)                  )                  SELECT emp\_hire\_date(t1.employee\_id, t1.hire\_date)                  BULK COLLECT INTO v\_hire\_dates                  FROM hire\_dates t1                  LEFT JOIN hire\_dates t2                  ON t1.employee\_id = t2.employee\_id AND t1.hire\_date < t2.hire\_date                  WHERE t2.hire\_date IS NULL;              END;          END IF;          SELECT emp\_hire\_day(employee\_id, TRUNC(hire\_date) - TRUNC(hire\_date, 'IW') + 1) AS hire\_day          BULK COLLECT INTO v\_hire\_days          FROM TABLE(v\_hire\_dates);          SELECT day\_nr          BULK COLLECT INTO v\_retval.empty\_days          FROM weekdays          LEFT JOIN TABLE(v\_hire\_days) hire\_days          ON weekdays.day\_nr = hire\_days.hire\_day          WHERE hire\_days.hire\_day IS NULL;          IF v\_retval.empty\_days.LAST = 7 THEN              RETURN v\_retval;          END IF;          WITH day\_counts AS (              SELECT hire\_day              FROM TABLE(v\_hire\_days)              GROUP BY hire\_day              ORDER BY COUNT(\*) DESC          )          SELECT hire\_day          INTO v\_retval.max\_day          FROM day\_counts          WHERE rownum = 1;          WITH emp\_to\_rank AS (              SELECT employee\_id, first\_name, last\_name, SYSDATE - hire\_date AS days\_worked              FROM employees              JOIN TABLE (v\_hire\_days)              USING (employee\_id)              WHERE hire\_day = v\_retval.max\_day              ORDER BY days\_worked DESC          )          SELECT e5\_employee(employee\_id, first\_name, last\_name, days\_worked, RANK() OVER (ORDER BY days\_worked DESC))          BULK COLLECT INTO v\_retval.max\_day\_employees          FROM emp\_to\_rank;          RETURN v\_retval;  END;  /  DECLARE      v\_dep\_name VARCHAR2(50) := &p\_dep\_name;      v\_use\_history BOOLEAN := &p\_use\_history;      v\_hire\_stats e5\_return;  BEGIN      dbms\_output.PUT\_LINE('Stats for department "'||v\_dep\_name||'". '|| CASE WHEN v\_use\_history THEN 'U' ELSE 'Not u' END||'sing job history.');      v\_hire\_stats := e5\_hire\_stats(v\_dep\_name, CASE WHEN v\_use\_history THEN 1 ELSE 0 END);      IF v\_hire\_stats.max\_day = 0 THEN          dbms\_output.PUT\_LINE('Department has no employees.');      ELSE          FOR v\_row IN (              SELECT day\_name FROM TABLE(v\_hire\_stats.empty\_days) ed              JOIN weekdays              ON weekdays.day\_nr = ed.column\_value          )          LOOP              dbms\_output.PUT\_LINE('No employees were hired on day '||v\_row.day\_name||'.');          END LOOP;          DECLARE              v\_day\_name VARCHAR2(50);          BEGIN              SELECT day\_name              INTO v\_day\_name              FROM weekdays              WHERE day\_nr = v\_hire\_stats.max\_day;              dbms\_output.PUT\_LINE('Weekday '||v\_day\_name||' has maximum number of hired employees.');          END;          DECLARE              v\_current\_rank BINARY\_INTEGER := 0;              v\_emp e5\_employee;          BEGIN              FOR i IN 1..v\_hire\_stats.max\_day\_employees.last              LOOP                  v\_emp := v\_hire\_stats.max\_day\_employees(i);                  IF v\_current\_rank != v\_emp.seniority\_rank                  THEN                      v\_current\_rank := v\_emp.seniority\_rank;                      dbms\_output.PUT\_LINE('Seniority rank '||v\_current\_rank||':');                  END IF;                  dbms\_output.PUT\_LINE(v\_emp.first\_name||' '||v\_emp.last\_name||' (ID '||v\_emp.employee\_id||', hired '||v\_emp.days\_hired||' days ago).');              END LOOP;          END;      END IF;  END;  DROP FUNCTION e5\_hire\_stats;  DROP TYPE emp\_hire\_days;  DROP TYPE emp\_hire\_day;  DROP TYPE emp\_hire\_dates;  DROP TYPE emp\_hire\_date;  DROP TYPE e5\_return;  DROP TYPE e5\_empty\_days;  DROP TYPE e5\_employees;  DROP TYPE e5\_employee;  DROP TABLE weekdays; |