# Creare tabel info\_restaurant

|  |
| --- |
| /\*  E7. Adaptați cerința  exercițiului  4  pentru  diagrama proiectului  prezentată la materia  Baze  de  Date  din  anul I. Rezolvați acest exercițiu în PL/SQL, folosind baza de date proprie.  \*/  /\*  4. (adaptat) a. Creați tabelul info\_restaurant cu următoarele coloane:  - id\_restaurant (codul restaurantului) – cheie primară  - numar\_angajati  - total\_plati  (suma alocată lunar pentru plata salariilor)  \*/  CREATE TABLE info\_restaurant (      id\_restaurant NUMBER(10) PRIMARY KEY,      numar\_angajati NUMBER(10) NOT NULL,      plati NUMBER(10) NOT NULL  ); |

# Inserare date

|  |
| --- |
| /\*   4. (adaptat) b. Introduceți date în tabelul creat anterior corespunzătoare informațiilor existente în schemă.  \*/  INSERT INTO info\_restaurant      /\* nu există restaurante fără angajați, dar dacă ar exista funcția ar funcționa corect cu ele \*/      SELECT id\_restaurant, COUNT(id\_angajat), SUM(COALESCE(salariu, 0)) AS count      FROM restaurant      LEFT JOIN angajat      USING (id\_restaurant)      GROUP BY id\_restaurant |

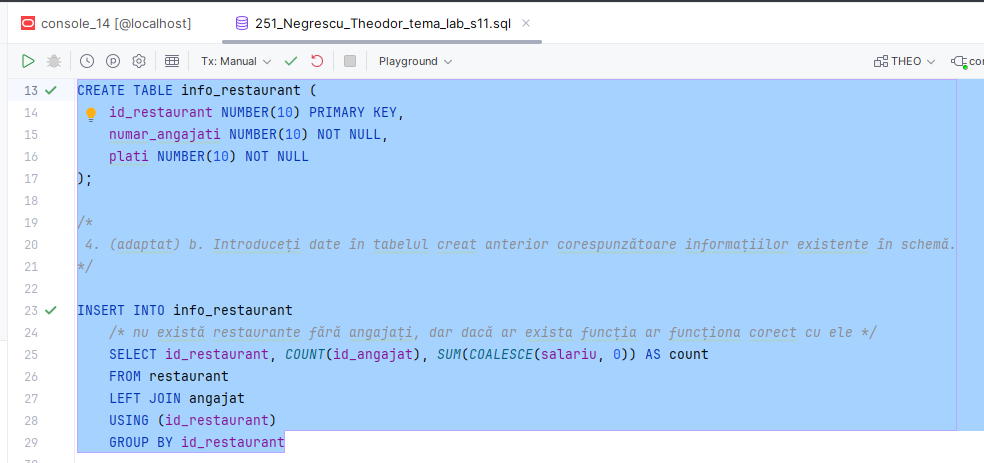
# Creare trigger

|  |
| --- |
| /\*   4. (adaptat) c.  Definiți  un  declanșator  care  va  actualiza  automat  câmpul  plati  atunci  când  se  introduce  un  nou angajat, respectiv se șterge angajatul sau se modifică salariul.  \*/  CREATE OR REPLACE PROCEDURE delete\_angajat(v\_rest restaurant.id\_restaurant%TYPE, v\_sal angajat.salariu%TYPE)  AS  BEGIN      UPDATE info\_restaurant      SET numar\_angajati = numar\_angajati - 1, plati = plati - v\_sal      WHERE id\_restaurant = v\_rest;  END;  CREATE OR REPLACE PROCEDURE add\_angajat(v\_rest restaurant.id\_restaurant%TYPE, v\_sal angajat.salariu%TYPE)  AS  BEGIN      UPDATE info\_restaurant      SET numar\_angajati = numar\_angajati + 1, plati = plati + v\_sal      WHERE id\_restaurant = v\_rest;  END;  CREATE OR REPLACE TRIGGER update\_info\_restaurant      BEFORE INSERT OR DELETE OR UPDATE ON angajat      FOR EACH ROW  BEGIN      IF DELETING OR UPDATING THEN          delete\_angajat(:OLD.id\_restaurant, :OLD.salariu);      END IF;      IF INSERTING OR UPDATING THEN          add\_angajat(:NEW.id\_restaurant, :NEW.salariu);      END IF;  END; |

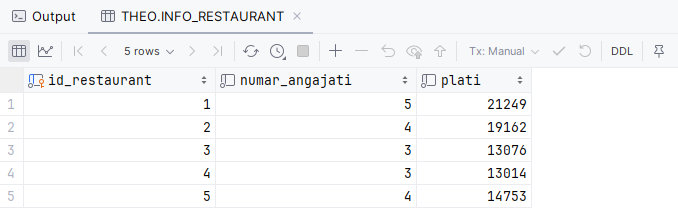
# Testare cod

|  |
| --- |
| SELECT \* FROM info\_restaurant;  UPDATE angajat  SET id\_restaurant = 2  WHERE id\_restaurant = 1 AND id\_angajat != 1;  DELETE FROM comanda  WHERE id\_restaurant = 4;  DELETE FROM angajat  WHERE id\_restaurant = 4;  INSERT INTO angajat (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu)  VALUES (5, NULL, 'MANAGER', 'test', SYSDATE, 1000000);  SELECT \* FROM info\_restaurant; |

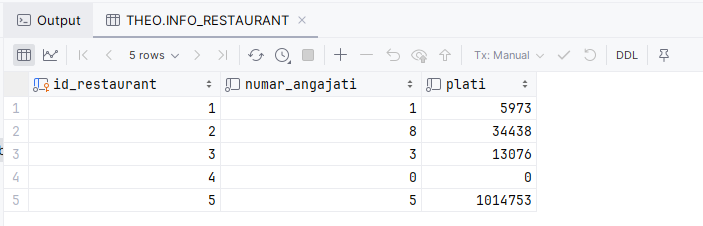
# Screenshot-uri

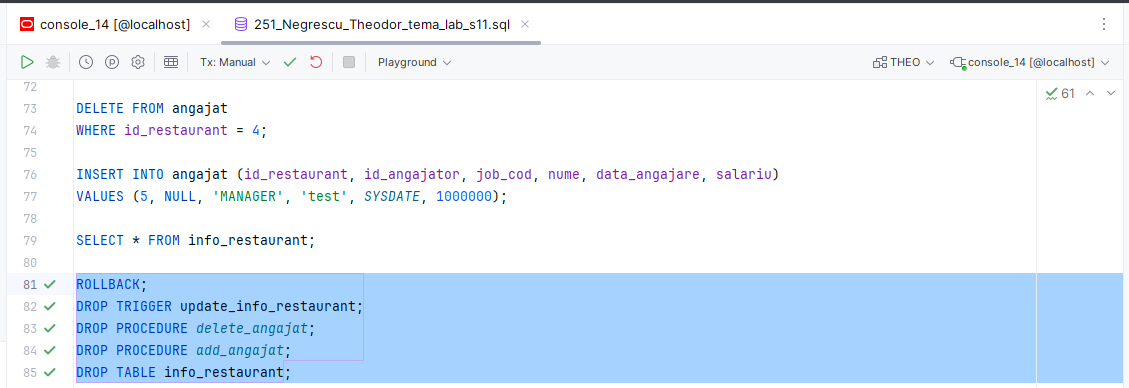












# Întregul fișier 251\_Negrescu\_Theodor\_tema\_lab\_s11.sql

|  |
| --- |
| /\*  E7. Adaptați cerința  exercițiului  4  pentru  diagrama proiectului  prezentată la materia  Baze  de  Date  din  anul I. Rezolvați acest exercițiu în PL/SQL, folosind baza de date proprie.  \*/  /\*  4. (adaptat) a. Creați tabelul info\_restaurant cu următoarele coloane:  - id\_restaurant (codul restaurantului) – cheie primară  - numar\_angajati  - total\_plati  (suma alocată lunar pentru plata salariilor)  \*/  CREATE TABLE info\_restaurant (      id\_restaurant NUMBER(10) PRIMARY KEY,      numar\_angajati NUMBER(10) NOT NULL,      plati NUMBER(10) NOT NULL  );  /\*   4. (adaptat) b. Introduceți date în tabelul creat anterior corespunzătoare informațiilor existente în schemă.  \*/  INSERT INTO info\_restaurant      /\* nu există restaurante fără angajați, dar dacă ar exista funcția ar funcționa corect cu ele \*/      SELECT id\_restaurant, COUNT(id\_angajat), SUM(COALESCE(salariu, 0)) AS count      FROM restaurant      LEFT JOIN angajat      USING (id\_restaurant)      GROUP BY id\_restaurant  /\*   4. (adaptat) c.  Definiți  un  declanșator  care  va  actualiza  automat  câmpul  plati  atunci  când  se  introduce  un  nou angajat, respectiv se șterge angajatul sau se modifică salariul.  \*/  CREATE OR REPLACE PROCEDURE delete\_angajat(v\_rest restaurant.id\_restaurant%TYPE, v\_sal angajat.salariu%TYPE)  AS  BEGIN      UPDATE info\_restaurant      SET numar\_angajati = numar\_angajati - 1, plati = plati - v\_sal      WHERE id\_restaurant = v\_rest;  END;  CREATE OR REPLACE PROCEDURE add\_angajat(v\_rest restaurant.id\_restaurant%TYPE, v\_sal angajat.salariu%TYPE)  AS  BEGIN      UPDATE info\_restaurant      SET numar\_angajati = numar\_angajati + 1, plati = plati + v\_sal      WHERE id\_restaurant = v\_rest;  END;  CREATE OR REPLACE TRIGGER update\_info\_restaurant      BEFORE INSERT OR DELETE OR UPDATE ON angajat      FOR EACH ROW  BEGIN      IF DELETING OR UPDATING THEN          delete\_angajat(:OLD.id\_restaurant, :OLD.salariu);      END IF;      IF INSERTING OR UPDATING THEN          add\_angajat(:NEW.id\_restaurant, :NEW.salariu);      END IF;  END;  SELECT \* FROM info\_restaurant;  UPDATE angajat  SET id\_restaurant = 2  WHERE id\_restaurant = 1 AND id\_angajat != 1;  DELETE FROM comanda  WHERE id\_restaurant = 4;  DELETE FROM angajat  WHERE id\_restaurant = 4;  INSERT INTO angajat (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu)  VALUES (5, NULL, 'MANAGER', 'test', SYSDATE, 1000000);  SELECT \* FROM info\_restaurant;  ROLLBACK;  DROP TRIGGER update\_info\_restaurant;  DROP PROCEDURE delete\_angajat;  DROP PROCEDURE add\_angajat;  DROP TABLE info\_restaurant; |