VIDEOGAME DATABASE

Proiect SGBD

Vasiliu Diana-Elena

Grupa 232, Anul II

Prezentare

Baza de date prezentată modelează informațiile necesare din cadrul unui magazin online de jocuri video. Această bază de date conține informații despre jocurile puse la vânzare de către magazin, precum și informații despre clienți și despre comenzile realizate de ei. O astfel de bază de date ar putea fi folosită în cadrul magazinului online Steam (https://store.steampowered.com/), după care a fost inspirat proiectul.

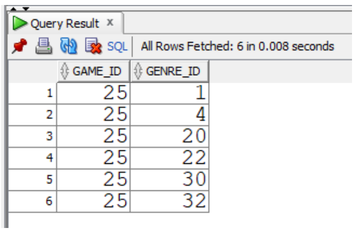
Baza de date este folosită pentru a stoca informații de bază despre angajații magazinului, precum și informații despre clienți și despre comenzi. Acest lucru este util pentru a avea un istoric al comenzilor, care este vizibil în contul fiecărui client, pentru un posibil sistem de recompensare a clienților (de exemplu, vouchere) ș.a. Baza de date stochează și informații despre jocurile vândute, pentru a fi prezentate în cadrul magazinului cu toate detaliile necesare pe care clientul le ia în vedere la cumpărarea unui joc.

Baza de date conține 19 tabele, din care 14 tabele sunt independente și 5 tabele sunt asociative.

Pentru a fi corectă, o bază de date trebuie să fie normalizată. Acest lucru este realizat în modelul de față până la forma normală 3.

Forma normală 1 (FN1) presupune ca fiecărui atribut să îi corespundă o valoare indivizibilă. De exemplu, modelul în cauză permite ca un joc să aibă mai multe genuri. Acest lucru se transpune în tabelul asociativ game_genre prin înregistrări multiple pentru id-ul jocului x, dar valori diferite corespunzătoare ale genurilor lui.

```sql
SELECT game_id, genre_id
FROM game_genre
WHERE game_id = 25
ORDER BY 2;
```



	GAME_ID	GENRE_ID
1	25	1
2	25	4
3	25	20
4	25	22
5	25	30
6	25	32

În exemplul de mai sus, relația a fost adusă în forma normală 1 prin înlocuirea înregistrării de tip (game_id = 25, genre_id = 1,4,20,22,30,32) cu mai multe înregistrări care au împărțit genre_id, creând valori indivizibile.

Forma normală 2 (FN2) presupune ca relația (tabelul) să fie în forma normală 1 (demonstrat adevărat mai sus), iar fiecare atribut care nu face parte din cheia primară să fie dependent de întreaga cheie primară. Dacă tabelul are cheia primară constituită dintr-un singur atribut sau dacă este artificială (este un identificator care nu are legătură directă cu datele din tabel – în general, numere), atunci este în FN2.

În modelul de față, majoritatea tabelelor au cheie primară artificială sau sunt tabele asociative cu singurele atribute în cheia primară (de exemplu, game_modes). Cel la care se pune întrebarea dacă este în FN2 este order_game.

O comandă poate să conțină mai multe jocuri. Această relație se transpune în tabelul order_game, care, pe lângă cheia primară, conține și date despre platforma pentru care a fost cumpărat jocul, cantitatea cumpărată și discount-ul. Atributul platform_id nu poate să depindă doar de game_id, întrucât nu se știe publisher-ul jocului cumpărat. Totodată, la o comandă trebuie să se știe platforma pentru care a fost cumpărat jocul, deci platform_id depinde și de game_id, și de order_id. Atributul quantity_id este o informație despre numărul de exemplare (jocuri) cumpărate, dar la comenzi diferite poate fi cumpărat același joc, diferind cantitatea, deci quiantity_id depinde de întreaga cheie primară. Discount-ul se poate aplica numai unui joc, nu poate exista independent, deci depinde de game_id. Totodată, discount-ul este o valoare care se paote schimba la fiecare comandă. Două comenzi diferite pot avea discount diferit pentru un joc, în funcție de oferta în care se află jocul. Deci, depinde și de order_id, adică de întreaga cheie primară.

Forma normală 3 (FN3) presupune ca tabelele să fie în forma normală 2 (demonstrat adevărat anterior), iar fiecare atribut care nu participă la cheia primară să fie direct dependentă de cheia primară. Această problemă apare doar la tabelele care au cel puțin două atribute care nu fac parte din cheia primară. După cum se poate observa, niciun tabel din modelul propus nu are atribute care să încalce regula proprie a formei normale 3.

Diagrama entitate-relație

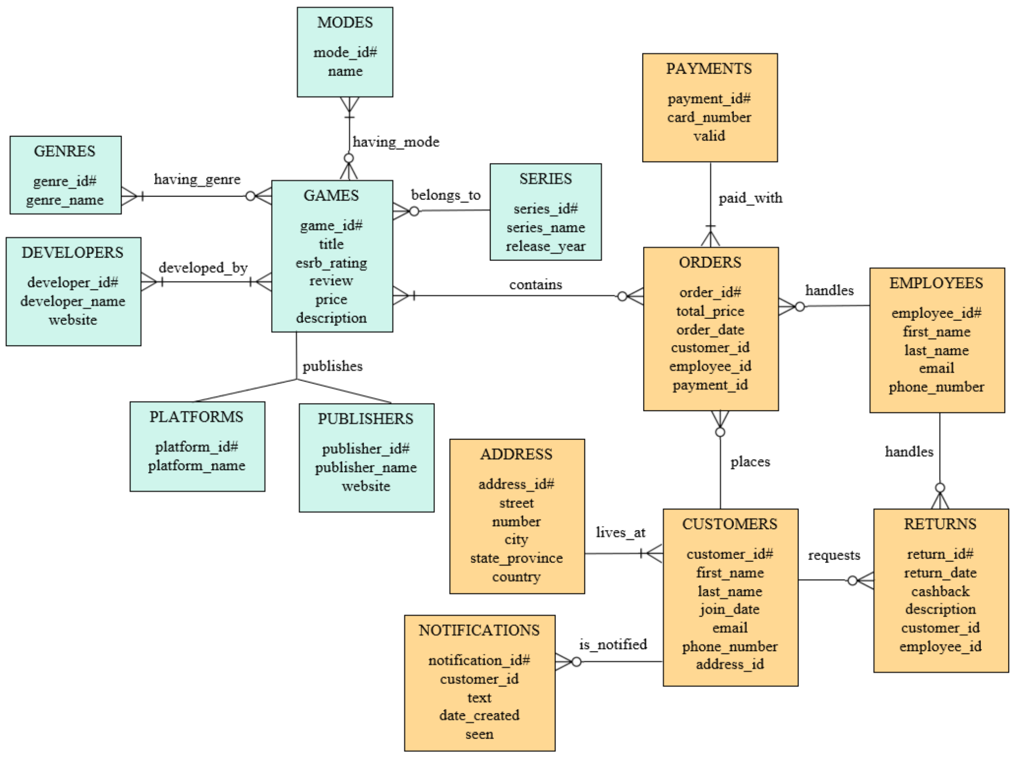
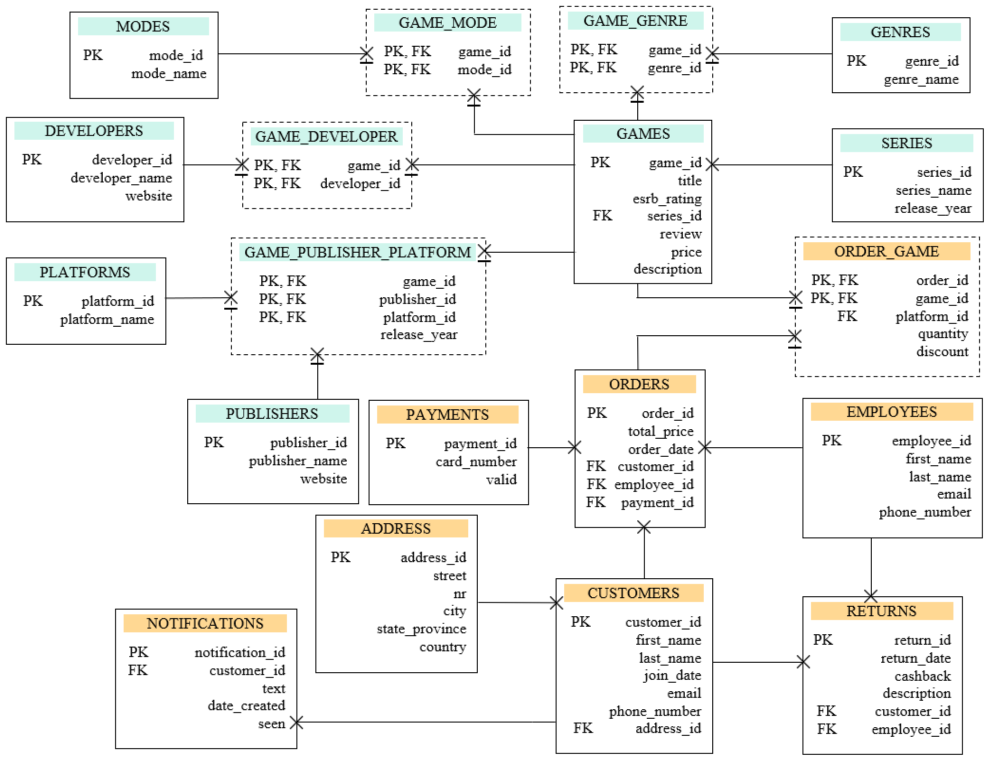
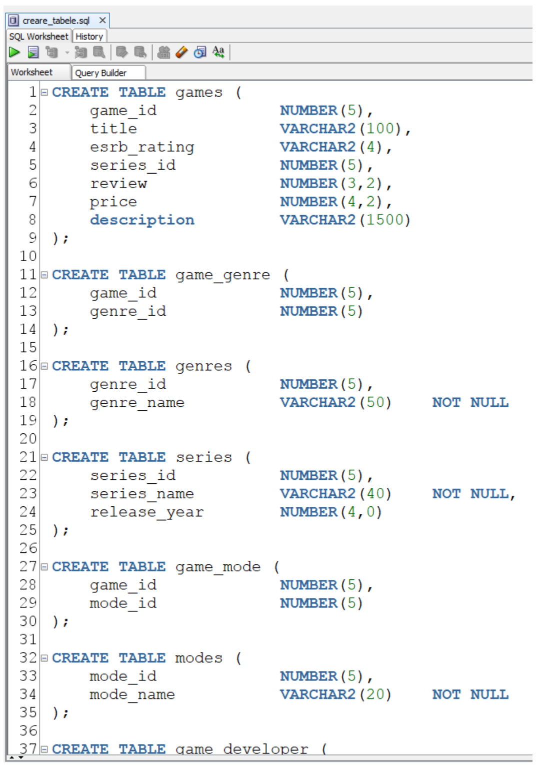


Diagrama conceptuală

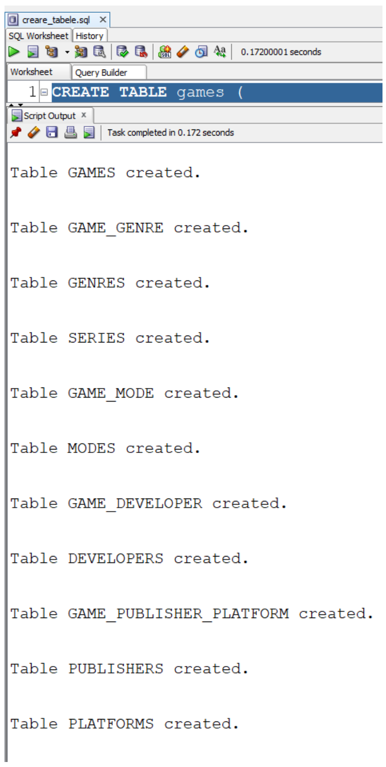


Crearea bazei de date

I. Crearea tabelelor



```sql
CREATE TABLE games (
    game_id             NUMBER(5),
    title               VARCHAR2(100),
    esrb_rating         VARCHAR2(4),
    series_id           NUMBER(5),
    review              NUMBER(3,2),
    price               NUMBER(4,2),
    description         VARCHAR2(1500)
);

CREATE TABLE game_genre (
    game_id             NUMBER(5),
    genre_id            NUMBER(5)
);

CREATE TABLE genres (
    genre_id            NUMBER(5),
    genre_name          VARCHAR2(50)    NOT NULL
);

CREATE TABLE series (
    series_id           NUMBER(5),
    series_name         VARCHAR2(40)    NOT NULL,
    release_year        NUMBER(4,0)
);

CREATE TABLE game_mode (
    game_id             NUMBER(5),
    mode_id             NUMBER(5)
);

CREATE TABLE modes (
    mode_id             NUMBER(5),
    mode_name           VARCHAR2(20)    NOT NULL
);

CREATE TABLE game_developer (
```

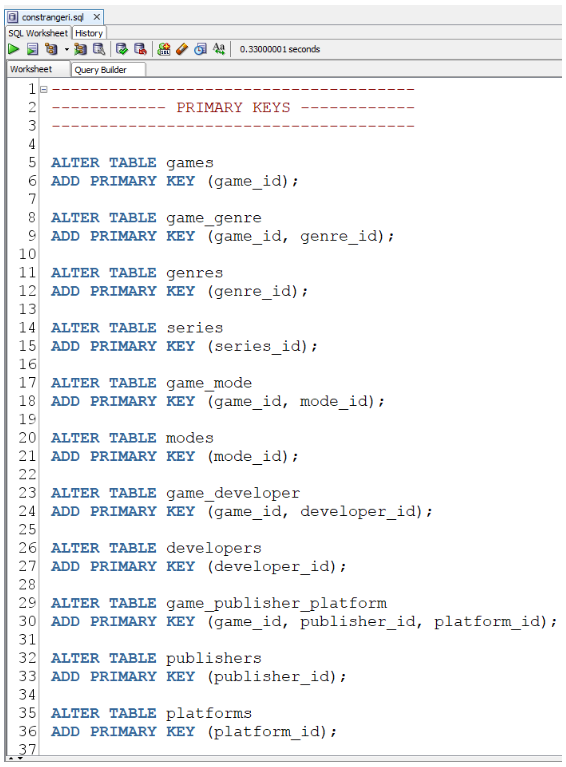


```sql
CREATE TABLE games (
```

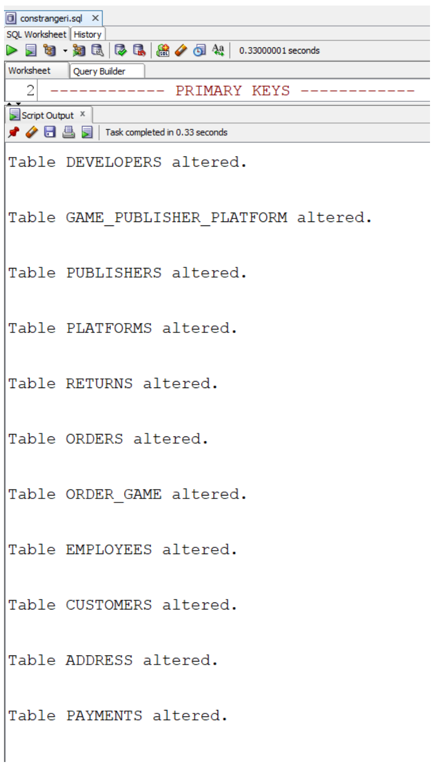
```
Table GAMES created.

Table GAME_GENRE created.

Table GENRES created.

Table SERIES created.

Table GAME_MODE created.

Table MODES created.

Table GAME_DEVELOPER created.

Table DEVELOPERS created.

Table GAME_PUBLISHER_PLATFORM created.

Table PUBLISHERS created.

Table PLATFORMS created.
```

II. Adăugarea constrângerilor

a. cheile primare



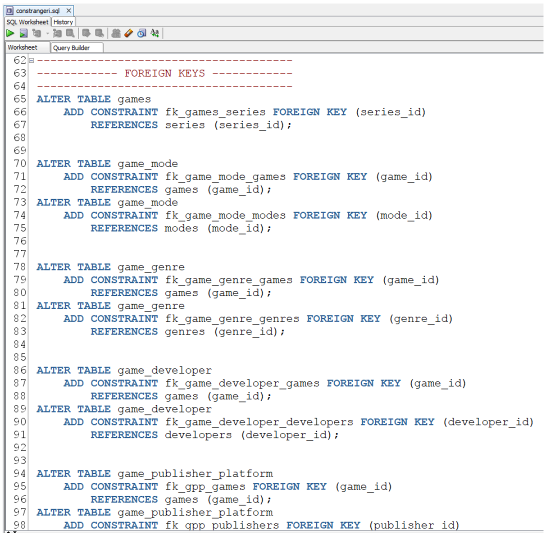
```sql
----------------------------------------
------------ PRIMARY KEYS ------------
----------------------------------------

ALTER TABLE games
ADD PRIMARY KEY (game_id);

ALTER TABLE game_genre
ADD PRIMARY KEY (game_id, genre_id);

ALTER TABLE genres
ADD PRIMARY KEY (genre_id);

ALTER TABLE series
ADD PRIMARY KEY (series_id);

ALTER TABLE game_mode
ADD PRIMARY KEY (game_id, mode_id);

ALTER TABLE modes
ADD PRIMARY KEY (mode_id);

ALTER TABLE game_developer
ADD PRIMARY KEY (game_id, developer_id);

ALTER TABLE developers
ADD PRIMARY KEY (developer_id);

ALTER TABLE game_publisher_platform
ADD PRIMARY KEY (game_id, publisher_id, platform_id);

ALTER TABLE publishers
ADD PRIMARY KEY (publisher_id);

ALTER TABLE platforms
ADD PRIMARY KEY (platform_id);
```



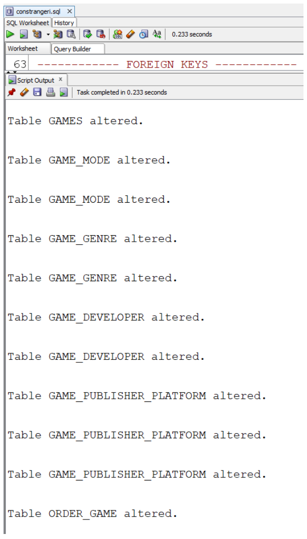
```sql
------------ PRIMARY KEYS ------------
```

```
Table DEVELOPERS altered.

Table GAME_PUBLISHER_PLATFORM altered.

Table PUBLISHERS altered.

Table PLATFORMS altered.

Table RETURNS altered.

Table ORDERS altered.

Table ORDER_GAME altered.

Table EMPLOYEES altered.

Table CUSTOMERS altered.

Table ADDRESS altered.

Table PAYMENTS altered.
```

b. cheile externe



```
---------------------------------------
------------ FOREIGN KEYS ------------
---------------------------------------
ALTER TABLE games
    ADD CONSTRAINT fk_games_series FOREIGN KEY (series_id)
        REFERENCES series (series_id);


ALTER TABLE game_mode
    ADD CONSTRAINT fk_game_mode_games FOREIGN KEY (game_id)
        REFERENCES games (game_id);
ALTER TABLE game_mode
    ADD CONSTRAINT fk_game_mode_modes FOREIGN KEY (mode_id)
        REFERENCES modes (mode_id);


ALTER TABLE game_genre
    ADD CONSTRAINT fk_game_genre_games FOREIGN KEY (game_id)
        REFERENCES games (game_id);
ALTER TABLE game_genre
    ADD CONSTRAINT fk_game_genre_genres FOREIGN KEY (genre_id)
        REFERENCES genres (genre_id);


ALTER TABLE game_developer
    ADD CONSTRAINT fk_game_developer_games FOREIGN KEY (game_id)
        REFERENCES games (game_id);
ALTER TABLE game_developer
    ADD CONSTRAINT fk_game_developer_developers FOREIGN KEY (developer_id)
        REFERENCES developers (developer_id);


ALTER TABLE game_publisher_platform
    ADD CONSTRAINT fk_gpp_games FOREIGN KEY (game_id)
        REFERENCES games (game_id);
ALTER TABLE game_publisher_platform
    ADD CONSTRAINT fk_gpp_publishers FOREIGN KEY (publisher_id)
```

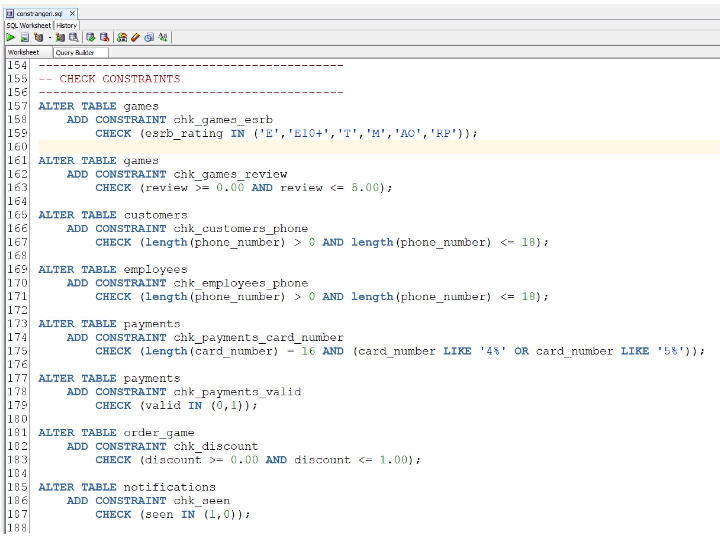


```
------------ FOREIGN KEYS ------------
```

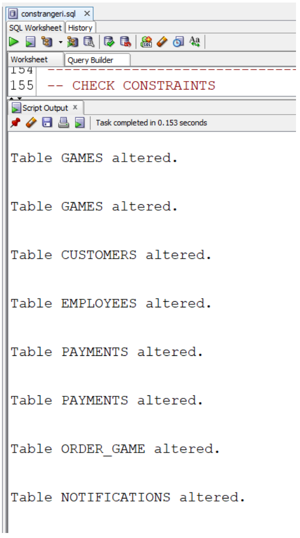
Task completed in 0.233 seconds

```
Table GAMES altered.

Table GAME_MODE altered.

Table GAME_MODE altered.

Table GAME_GENRE altered.

Table GAME_GENRE altered.

Table GAME_DEVELOPER altered.

Table GAME_DEVELOPER altered.

Table GAME_PUBLISHER_PLATFORM altered.

Table GAME_PUBLISHER_PLATFORM altered.

Table GAME_PUBLISHER_PLATFORM altered.

Table ORDER_GAME altered.
```

c. constrângeri de tip CHECK



```
-------------------------------------------
-- CHECK CONSTRAINTS
-------------------------------------------
ALTER TABLE games
    ADD CONSTRAINT chk_games_esrb
        CHECK (esrb_rating IN ('E','E10+','T','M','AO','RP'));

ALTER TABLE games
    ADD CONSTRAINT chk_games_review
        CHECK (review >= 0.00 AND review <= 5.00);

ALTER TABLE customers
    ADD CONSTRAINT chk_customers_phone
        CHECK (length(phone_number) > 0 AND length(phone_number) <= 18);

ALTER TABLE employees
    ADD CONSTRAINT chk_employees_phone
        CHECK (length(phone_number) > 0 AND length(phone_number) <= 18);

ALTER TABLE payments
    ADD CONSTRAINT chk_payments_card_number
        CHECK (length(card_number) = 16 AND (card_number LIKE '4%' OR card_number LIKE '5%'));

ALTER TABLE payments
    ADD CONSTRAINT chk_payments_valid
        CHECK (valid IN (0,1));

ALTER TABLE order_game
    ADD CONSTRAINT chk_discount
        CHECK (discount >= 0.00 AND discount <= 1.00);

ALTER TABLE notifications
    ADD CONSTRAINT chk_seen
        CHECK (seen IN (1,0));
```

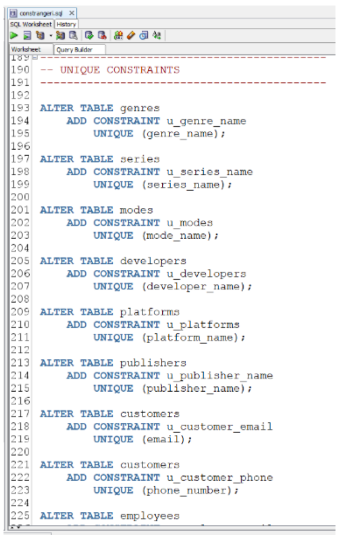


```
-- CHECK CONSTRAINTS
```

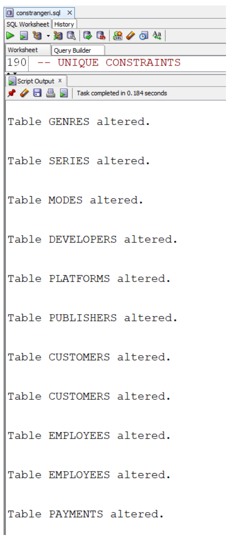
Task completed in 0.153 seconds

```
Table GAMES altered.

Table GAMES altered.

Table CUSTOMERS altered.

Table EMPLOYEES altered.

Table PAYMENTS altered.

Table PAYMENTS altered.

Table ORDER_GAME altered.

Table NOTIFICATIONS altered.
```

d. constrângeri de tip UNIQUE



```
-------------------------------------------
-- UNIQUE CONSTRAINTS
-------------------------------------------

ALTER TABLE genres
    ADD CONSTRAINT u_genre_name
        UNIQUE (genre_name);

ALTER TABLE series
    ADD CONSTRAINT u_series_name
        UNIQUE (series_name);

ALTER TABLE modes
    ADD CONSTRAINT u_modes
        UNIQUE (mode_name);

ALTER TABLE developers
    ADD CONSTRAINT u_developers
        UNIQUE (developer_name);

ALTER TABLE platforms
    ADD CONSTRAINT u_platforms
        UNIQUE (platform_name);

ALTER TABLE publishers
    ADD CONSTRAINT u_publisher_name
        UNIQUE (publisher_name);

ALTER TABLE customers
    ADD CONSTRAINT u_customer_email
        UNIQUE (email);

ALTER TABLE customers
    ADD CONSTRAINT u_customer_phone
        UNIQUE (phone_number);

ALTER TABLE employees
```



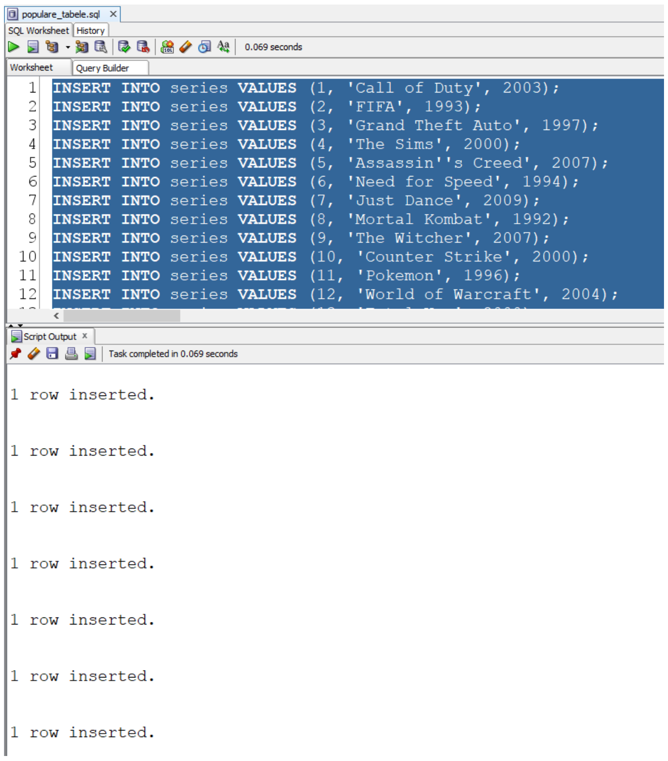
```
-- UNIQUE CONSTRAINTS
```

Task completed in 0.184 seconds

```
Table GENRES altered.

Table SERIES altered.

Table MODES altered.

Table DEVELOPERS altered.

Table PLATFORMS altered.

Table PUBLISHERS altered.

Table CUSTOMERS altered.

Table CUSTOMERS altered.

Table EMPLOYEES altered.

Table EMPLOYEES altered.

Table PAYMENTS altered.
```

- constrângerile de tip NOT NULL și DEFAULT au fost adăugate la crearea tabelelor

Introducerea informațiilor

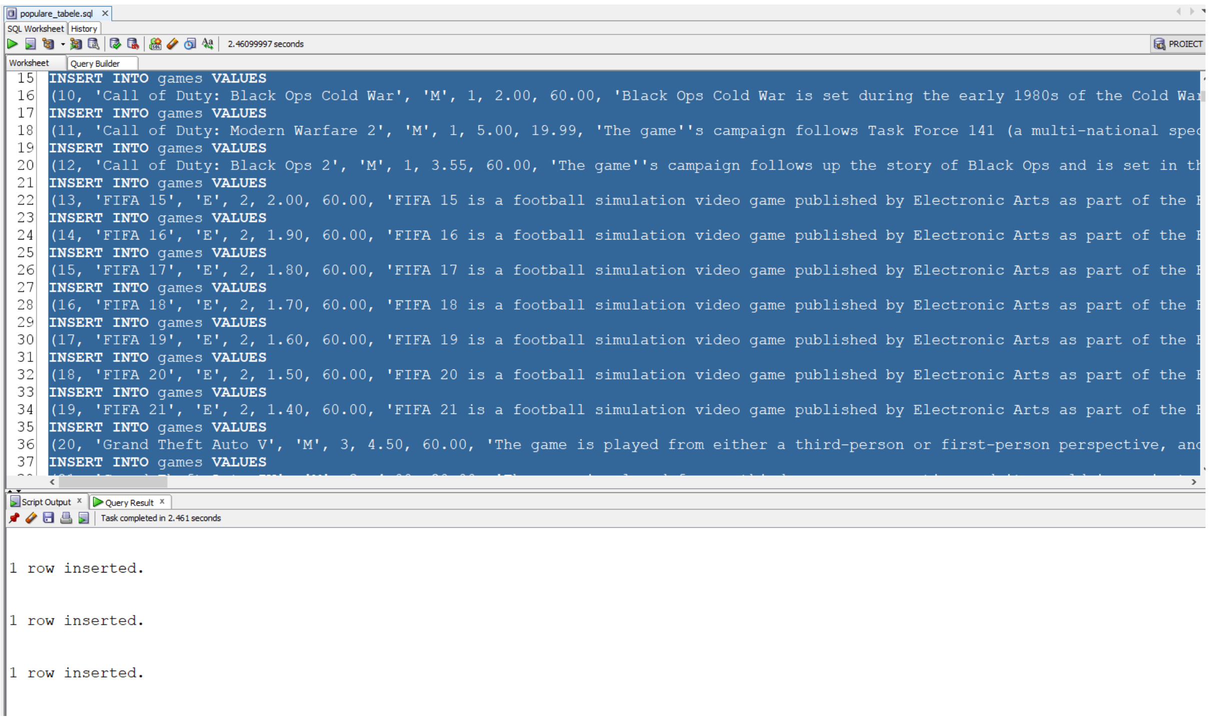


populare_tabele.sql | SQL Worksheet | History | 0.069 seconds | Worksheet | Query Builder

```sql
INSERT INTO series VALUES (1, 'Call of Duty', 2003);
INSERT INTO series VALUES (2, 'FIFA', 1993);
INSERT INTO series VALUES (3, 'Grand Theft Auto', 1997);
INSERT INTO series VALUES (4, 'The Sims', 2000);
INSERT INTO series VALUES (5, 'Assassin''s Creed', 2007);
INSERT INTO series VALUES (6, 'Need for Speed', 1994);
INSERT INTO series VALUES (7, 'Just Dance', 2009);
INSERT INTO series VALUES (8, 'Mortal Kombat', 1992);
INSERT INTO series VALUES (9, 'The Witcher', 2007);
INSERT INTO series VALUES (10, 'Counter Strike', 2000);
INSERT INTO series VALUES (11, 'Pokemon', 1996);
INSERT INTO series VALUES (12, 'World of Warcraft', 2004);
```

Script Output | Task completed in 0.069 seconds

```
1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```



populare_tabele.sql | SQL Worksheet | History | 2.46099997 seconds | PROIECT | Worksheet | Query Builder

```sql
INSERT INTO games VALUES
(10, 'Call of Duty: Black Ops Cold War', 'M', 1, 2.00, 60.00, 'Black Ops Cold War is set during the early 1980s of the Cold Wa
INSERT INTO games VALUES
(11, 'Call of Duty: Modern Warfare 2', 'M', 1, 5.00, 19.99, 'The game''s campaign follows Task Force 141 (a multi-national spe
INSERT INTO games VALUES
(12, 'Call of Duty: Black Ops 2', 'M', 1, 3.55, 60.00, 'The game''s campaign follows up the story of Black Ops and is set in t
INSERT INTO games VALUES
(13, 'FIFA 15', 'E', 2, 2.00, 60.00, 'FIFA 15 is a football simulation video game published by Electronic Arts as part of the
INSERT INTO games VALUES
(14, 'FIFA 16', 'E', 2, 1.90, 60.00, 'FIFA 16 is a football simulation video game published by Electronic Arts as part of the
INSERT INTO games VALUES
(15, 'FIFA 17', 'E', 2, 1.80, 60.00, 'FIFA 17 is a football simulation video game published by Electronic Arts as part of the
INSERT INTO games VALUES
(16, 'FIFA 18', 'E', 2, 1.70, 60.00, 'FIFA 18 is a football simulation video game published by Electronic Arts as part of the
INSERT INTO games VALUES
(17, 'FIFA 19', 'E', 2, 1.60, 60.00, 'FIFA 19 is a football simulation video game published by Electronic Arts as part of the
INSERT INTO games VALUES
(18, 'FIFA 20', 'E', 2, 1.50, 60.00, 'FIFA 20 is a football simulation video game published by Electronic Arts as part of the
INSERT INTO games VALUES
(19, 'FIFA 21', 'E', 2, 1.40, 60.00, 'FIFA 21 is a football simulation video game published by Electronic Arts as part of the
INSERT INTO games VALUES
(20, 'Grand Theft Auto V', 'M', 3, 4.50, 60.00, 'The game is played from either a third-person or first-person perspective, an
INSERT INTO games VALUES
```

Script Output | Query Result | Task completed in 2.461 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.
```

Rezolvarea exercițiilor propuse în PL/SQL

Problema 6. **Cerință:**

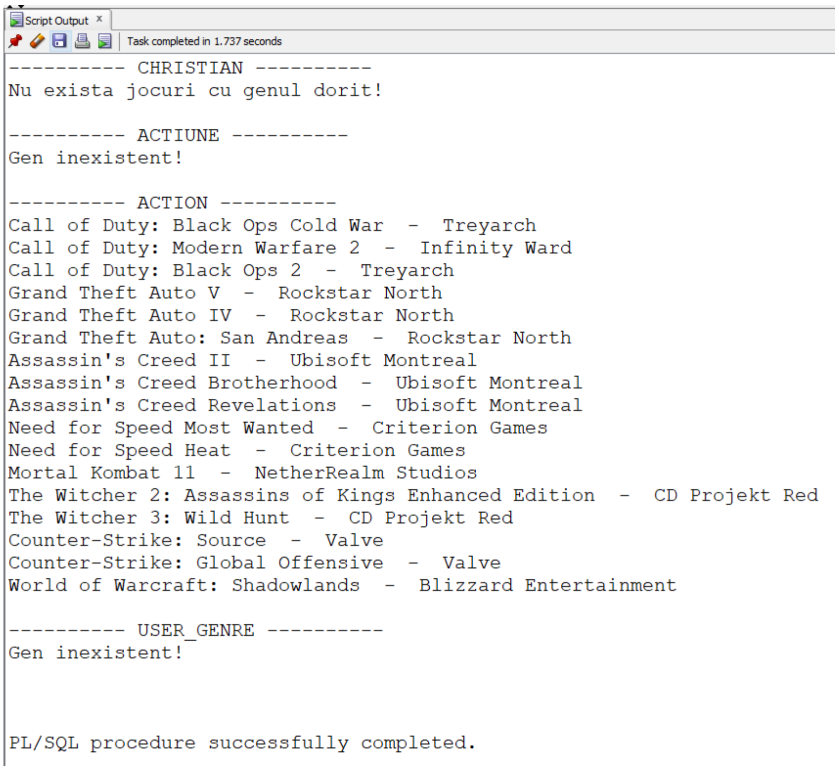
Afișați titlurile și numele developer-ului pentru fiecare dintre jocurile care au un gen specificat.

Pentru această problemă, am creat o procedură stocată care primește ca parametru numele genului dorit și folosește un tablou imbricat pentru reținerea id-urilor jocurilor cu acest gen, pentru a putea căuta mai ușor developerii fiecăruia.

```
CREATE OR REPLACE
    PROCEDURE game_developer_list ( v_genre_name  genres.genre_name%TYPE )
AS
    TYPE    t_game_id   IS TABLE OF     games.game_id%TYPE;
    TYPE    t_title     IS TABLE OF     games.title%TYPE;
    TYPE    t_developer IS TABLE OF     developers.developer_name%TYPE;
    v_genre_id          game_genre.genre_id%TYPE;
    v_game_id           t_game_id       := t_game_id();
    v_title             t_title         := t_title();
    v_devname           t_developer     := t_developer();
    e_no_games          EXCEPTION;
BEGIN

    dbms_output.put_line('---------- ' || UPPER(v_genre_name) || ' ----------');

    SELECT genre_id
    INTO v_genre_id
    FROM genres
    WHERE INITCAP(genre_name) = INITCAP(v_genre_name);

    SELECT game_id
    BULK COLLECT INTO v_game_id
    FROM game_genre
    WHERE genre_id = v_genre_id;

    IF v_game_id.COUNT != 0 THEN
        FOR i IN v_game_id.FIRST .. v_game_id.LAST LOOP

            SELECT title, developer_name
            BULK COLLECT INTO v_title, v_devname
            FROM games
            JOIN game_developer USING (game_id)
            JOIN developers USING (developer_id)
            WHERE game_id = v_game_id(i);

            FOR j IN v_devname.FIRST .. v_devname.LAST LOOP
               dbms_output.put_line(v_title(j) || '  -  ' || v_devname(j));
            END LOOP;

        END LOOP;
    ELSE
        RAISE e_no_games;
    END IF;

    dbms_output.new_line;
EXCEPTION
    WHEN e_no_games THEN
        dbms_output.put_line('Nu exista jocuri cu genul dorit!');
        dbms_output.new_line;
    WHEN NO_DATA_FOUND THEN
        dbms_output.put_line('Gen inexistent!');
        dbms_output.new_line;
    WHEN OTHERS THEN
        dbms_output.put_line('Alta eroare! - ' || SQLERRM);
        dbms_output.new_line;
END game_developer_list;
/
```

Am apelat procedura într-un bloc PL/SQL, cu diferiți parametri, pentru a evidenția tratarea excepțiilor.

```
BEGIN
    game_developer_list('Christian');
    game_developer_list('Actiune');
    game_developer_list('Action');
    game_developer_list('&p_gen');
END;
/
```



```
---------- CHRISTIAN ----------
Nu exista jocuri cu genul dorit!

---------- ACTIUNE ----------
Gen inexistent!

---------- ACTION ----------
Call of Duty: Black Ops Cold War  -  Treyarch
Call of Duty: Modern Warfare 2  -  Infinity Ward
Call of Duty: Black Ops 2  -  Treyarch
Grand Theft Auto V  -  Rockstar North
Grand Theft Auto IV  -  Rockstar North
Grand Theft Auto: San Andreas  -  Rockstar North
Assassin's Creed II  -  Ubisoft Montreal
Assassin's Creed Brotherhood  -  Ubisoft Montreal
Assassin's Creed Revelations  -  Ubisoft Montreal
Need for Speed Most Wanted  -  Criterion Games
Need for Speed Heat  -  Criterion Games
Mortal Kombat 11  -  NetherRealm Studios
The Witcher 2: Assassins of Kings Enhanced Edition  -  CD Projekt Red
The Witcher 3: Wild Hunt  -  CD Projekt Red
Counter-Strike: Source  -  Valve
Counter-Strike: Global Offensive  -  Valve
World of Warcraft: Shadowlands  -  Blizzard Entertainment

---------- USER_GENRE ----------
Gen inexistent!



PL/SQL procedure successfully completed.
```

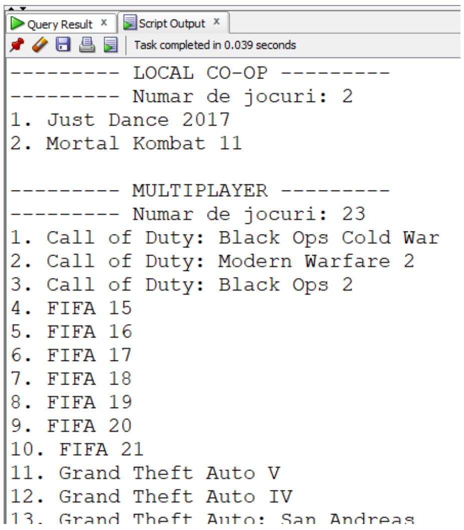
Problema 7. **Cerință:**

Pentru fiecare mod, afișați, în ordine alfabetică, numele modului, numărul de jocuri care dispun de acest mod și o listă cu aceste jocuri, numerotate de la 1.

Pentru această problemă, am creat o procedură stocată în care am folosit un ciclu cursor cu subcereri pentru a itera prin lista de moduri (pentru afișarea titlurilor și informațiilor pe rând pentru fiecare mod) și un ciclu cursor cu subcereri pentru a itera prin lista de jocuri care au acest mod (pentru afișarea titlurilor sub formă de listă).

```
CREATE OR REPLACE
    PROCEDURE show_modes_game_list
AS
    v_pos        NUMBER;
    v_number     NUMBER;
BEGIN

    FOR v_mode IN ( SELECT mode_id, mode_name
                    FROM modes
                    ORDER BY mode_name )
    LOOP
        dbms_output.put_line('--------- ' || UPPER(v_mode.mode_name) || ' ---------');

        SELECT COUNT(*)
        INTO v_number
        FROM game_mode
        WHERE mode_id = v_mode.mode_id;

        dbms_output.put_line('--------- Numar de jocuri: ' || v_number);

        v_pos := 1;
        FOR v_title IN ( SELECT title
                         FROM games
                         JOIN game_mode USING (game_id)
                         WHERE mode_id = v_mode.mode_id )
        LOOP
            dbms_output.put_line(v_pos || '. ' || v_title.title);
            v_pos := v_pos + 1;
        END LOOP;

        dbms_output.new_line;
    END LOOP;
END show_modes_game_list;
/
```

```
BEGIN
    show_modes_game_list;
END;
/
```



```
--------- LOCAL CO-OP ---------
--------- Numar de jocuri: 2
1. Just Dance 2017
2. Mortal Kombat 11

--------- MULTIPLAYER ---------
--------- Numar de jocuri: 23
1. Call of Duty: Black Ops Cold War
2. Call of Duty: Modern Warfare 2
3. Call of Duty: Black Ops 2
4. FIFA 15
5. FIFA 16
6. FIFA 17
7. FIFA 18
8. FIFA 19
9. FIFA 20
10. FIFA 21
11. Grand Theft Auto V
12. Grand Theft Auto IV
13. Grand Theft Auto: San Andreas
```

Problema 8. **Cerință:**

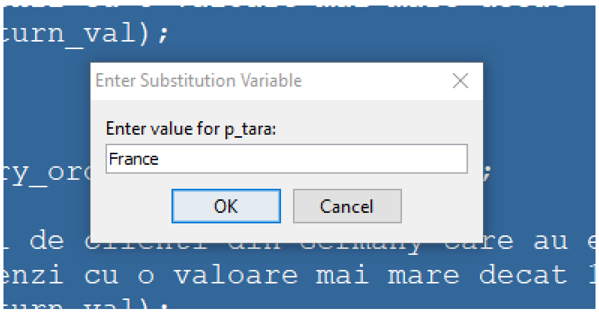
Calculați numărul de clienți care locuiesc într-o țară specificată și care au plasat comenzi cu o valoare totală mai mare decât o valoare dată.

Pentru această problemă, am creat o funcție stocată care returnează un număr și care folosește tabelele ”Address”, ”Customers” și ”Orders”.

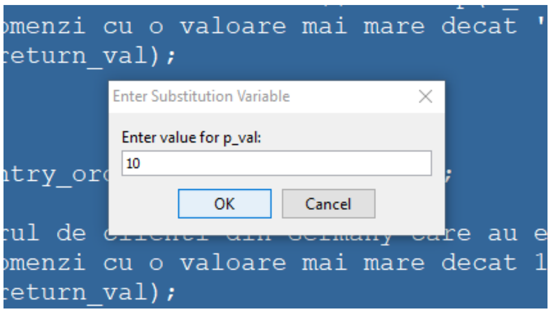
```
CREATE OR REPLACE
    FUNCTION nr_clients_country_orderval ( v_country_name   address.country%TYPE,
                                           v_over           VARCHAR2 )
    RETURN NUMBER
AS
    TYPE    t_customer_id   IS TABLE OF customers.customer_id%TYPE;
    v_customer_id       t_customer_id   := t_customer_id();
    v_order_nr          NUMBER;
    v_number            NUMBER          := 0;
    v_limit             orders.total_price%TYPE;
    e_no_country        EXCEPTION;
BEGIN

    v_limit := TO_NUMBER(v_over);

    SELECT customer_id
    BULK COLLECT INTO v_customer_id
    FROM customers
    WHERE address_id IN ( SELECT address_id
                          FROM address
                          WHERE LOWER(country) = LOWER(v_country_name) );

    IF v_customer_id.COUNT = 0 THEN
        RAISE e_no_country;
    END IF;

    FOR i IN v_customer_id.FIRST .. v_customer_id.LAST LOOP

        SELECT COUNT(DISTINCT customer_id)
        INTO v_order_nr
        FROM orders
        WHERE customer_id = v_customer_id(i)
        AND total_price >= v_limit;

        v_number := v_number + v_order_nr;
    END LOOP;

    RETURN v_number;

EXCEPTION
    WHEN e_no_country THEN
        dbms_output.put_line('Nu exista clienti cu locuinta in ' || v_country_name || '!');
        RETURN -1;
    WHEN VALUE_ERROR THEN
        dbms_output.put_line('Format gresit pentru valoarea ''' || v_over || '''!');
        RETURN -6502;
    WHEN OTHERS THEN
        dbms_output.put_line('Alta eroare! - ' || SQLERRM || ' - ' || SQLCODE);
        RETURN -20005;

END nr_clients_country_orderval;
/
```

Am apelat funcția într-un bloc PL/SQL cu 4 variante de valori pentru a evidenția tratarea excepțiilor.

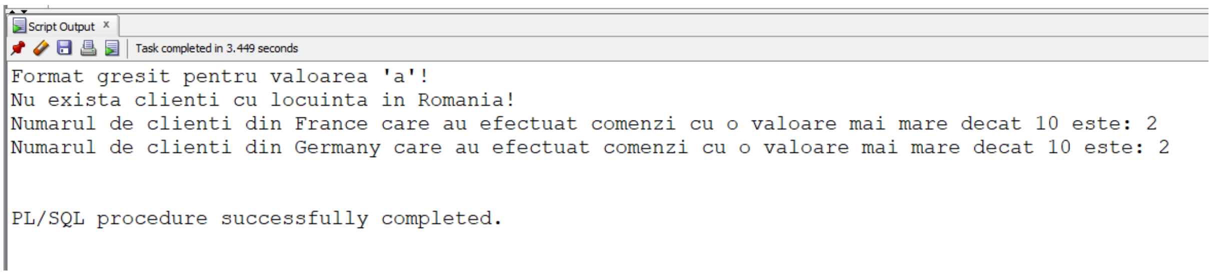
```sql
DECLARE
    v_return_val    NUMBER;
    v_tara          VARCHAR2(200)  := '&p_tara';
    v_val           VARCHAR2(200)  := '&p_val';
BEGIN
    -- test 1
    v_return_val := nr_clients_country_orderval ('Germany', 'a');
    IF v_return_val >= 0 THEN
        dbms_output.put_line('Numarul de clienti din Germany care au efectuat '
                             || 'comenzi cu o valoare mai mare decat 10 este: '
                             || v_return_val);
    END IF;

    -- test 2
    v_return_val := nr_clients_country_orderval ('Romania', 100);
    IF v_return_val >= 0 THEN
        dbms_output.put_line('Numarul de clienti din Romania care au efectuat '
                             || 'comenzi cu o valoare mai mare decat 100 este: '
                             || v_return_val);
    END IF;

    -- test 3
    v_return_val := nr_clients_country_orderval (v_tara, v_val);
    IF v_return_val >= 0 THEN
        dbms_output.put_line('Numarul de clienti din ' || initcap(v_tara) || ' care au efectuat '
                             || 'comenzi cu o valoare mai mare decat ' || v_val || ' este: '
                             || v_return_val);
    END IF;

    -- test 4
    v_return_val := nr_clients_country_orderval ('Germany', 10);
    IF v_return_val >= 0 THEN
        dbms_output.put_line('Numarul de clienti din Germany care au efectuat '
                             || 'comenzi cu o valoare mai mare decat 10 este: '
                             || v_return_val);
    END IF;
END;
/
```



Enter value for p_tara: France



Enter value for p_val: 10



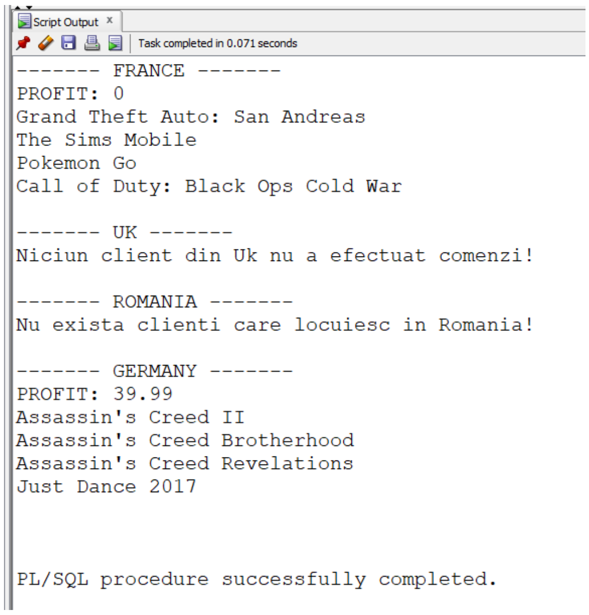
```
Format gresit pentru valoarea 'a'!
Nu exista clienti cu locuinta in Romania!
Numarul de clienti din France care au efectuat comenzi cu o valoare mai mare decat 10 este: 2
Numarul de clienti din Germany care au efectuat comenzi cu o valoare mai mare decat 10 este: 2


PL/SQL procedure successfully completed.
```

Problema 9. **Cerință:**

Afișați toate jocurile (titlul lor) care au fost cumpărate de clienți care trăiesc într-o țară specificată și câștigurile din acea țară.

```
CREATE OR REPLACE
    PROCEDURE show_ordered_games_country ( v_country_name     address.country%TYPE )
AS
    TYPE    t_customers     IS TABLE OF     customers.customer_id%TYPE;
    TYPE    t_games         IS TABLE OF     games.game_id%TYPE;
    TYPE    t_orders        IS TABLE OF     orders.order_id%TYPE;
    TYPE    t_price         IS TABLE OF     orders.total_price%TYPE;
    v_customer_id           t_customers     := t_customers();
    v_game_temp             t_games         := t_games();
    v_game_id               t_games         := t_games();
    v_order_temp            t_orders        := t_orders();
    v_order_id              t_orders        := t_orders();
    v_price                 t_price         := t_price();
    v_title                 games.title%TYPE;
    v_profit                NUMBER;
    e_no_country            EXCEPTION;
    e_no_orders             EXCEPTION;
BEGIN

    dbms_output.put_line('------- ' || UPPER(v_country_name) || ' -------');

    SELECT customer_id
    BULK COLLECT INTO v_customer_id
    FROM customers
    WHERE address_id IN ( SELECT address_id
                          FROM address
                          WHERE LOWER(country) = LOWER(v_country_name));

    IF v_customer_id.COUNT = 0 THEN
        RAISE e_no_country;
    END IF;

    v_profit := 0;

    FOR i IN v_customer_id.FIRST .. v_customer_id.LAST LOOP
        SELECT order_id, total_price
        BULK COLLECT INTO v_order_temp, v_price
        FROM orders
        WHERE customer_id = v_customer_id(i);

        IF v_order_temp.COUNT != 0 THEN
            FOR j IN v_order_temp.FIRST .. v_order_temp.LAST LOOP
                v_order_id.EXTEND;
                v_order_id( v_order_id.COUNT ) := v_order_temp(j);
                v_profit := v_profit + v_price(j);
            END LOOP;
        END IF;

        SELECT cashback
        BULK COLLECT INTO v_price
        FROM returns
        WHERE customer_id = v_customer_id(i);

        IF v_price.COUNT != 0 THEN
            FOR j IN v_price.FIRST .. v_price.LAST LOOP
                v_profit := v_profit - v_price(j);
            END LOOP;
        END IF;
    END LOOP;

    IF v_order_id.COUNT = 0 THEN
        RAISE e_no_orders;
    END IF;
    dbms_output.put_line('PROFIT: '  || v_profit);

    FOR i IN v_order_id.FIRST .. v_order_id.LAST LOOP

        SELECT game_id
        BULK COLLECT INTO v_game_temp
        FROM order_game
        WHERE order_id = v_order_id(i);

        FOR j IN v_game_temp.FIRST .. v_game_temp.LAST LOOP
            v_game_id.EXTEND;
            v_game_id( v_game_id.COUNT ) := v_game_temp(j);
        END LOOP;

    END LOOP;

    v_game_id := SET(v_game_id);

    FOR i IN v_game_id.FIRST .. v_game_id.LAST LOOP
        SELECT title
        INTO v_title
        FROM games
        WHERE game_id = v_game_id(i);

        dbms_output.put_line(v_title);
    END LOOP;

    dbms_output.new_line;

EXCEPTION
    WHEN e_no_country THEN
        dbms_output.put_line('Nu exista clienti care locuiesc in ' ||
                                          INITCAP(v_country_name)  || '!');
        dbms_output.new_line;
    WHEN e_no_orders THEN
        dbms_output.put_line('Niciun client din ' || INITCAP(v_country_name) ||
                                          ' nu a efectuat comenzi!');
        dbms_output.new_line;
    WHEN OTHERS THEN
        dbms_output.put_line('Alta eroare! - ' || SQLERRM);
        dbms_output.new_line;

END show_ordered_games_country;
/
```

```
BEGIN
    show_ordered_games_country('France');
    show_ordered_games_country('UK');
    show_ordered_games_country('Romania');
    show_ordered_games_country('Germany');
END;
/
```


Script Output ×
Task completed in 0.071 seconds

```
------- FRANCE -------
PROFIT: 0
Grand Theft Auto: San Andreas
The Sims Mobile
Pokemon Go
Call of Duty: Black Ops Cold War

------- UK -------
Niciun client din Uk nu a efectuat comenzi!

------- ROMANIA -------
Nu exista clienti care locuiesc in Romania!

------- GERMANY -------
PROFIT: 39.99
Assassin's Creed II
Assassin's Creed Brotherhood
Assassin's Creed Revelations
Just Dance 2017



PL/SQL procedure successfully completed.
```

Problema 10. **Cerință:**

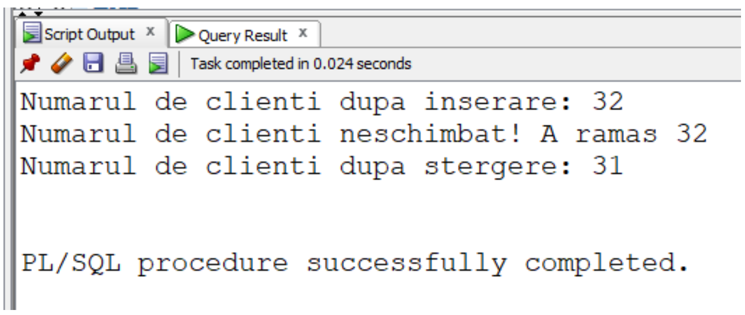
Să se afișeze numărul de clienți după fiecare operație efectuată asupra datelor lor (inserarea unui client nou, actualizarea datelor unui client sau ștergerea unui client).

Pentru rezolvarea acestei probleme, am creat un trigger LMD la nivel de comandă, care se declanșează după una dintre comenzile INSERT, UPDATE sau DELETE efectuate asupra tabelului ”CUSTOMERS”. Am calculat numărul de clienți după încheierea comenzii (fiind trigger AFTER), apoi, în funcție de comandă, am afișat un mesaj corespunzător.

```
CREATE OR REPLACE
    TRIGGER show_number_of_customers
AFTER INSERT OR UPDATE OR DELETE ON customers
DECLARE
    v_nr        NUMBER;
BEGIN

    SELECT COUNT(*)
    INTO v_nr
    FROM customers;

    IF INSERTING THEN
        dbms_output.put_line('Numarul de clienti dupa inserare: ' || v_nr);
    ELSIF UPDATING THEN
        dbms_output.put_line('Numarul de clienti neschimbat! A ramas ' || v_nr);
    ELSIF DELETING THEN
        dbms_output.put_line('Numarul de clienti dupa stergere: ' || v_nr);
    END IF;
END;
/
```

Am creat un bloc PL/SQL care inserează, actualizează și șterge un client din tabel, pentru evidențierea declanșării trigger-ului.

```
BEGIN
    INSERT INTO customers VALUES
    (1031, 'Alan', 'Turing', '06-MAY-2019', 'alanturing@gmail.com', '074 2356 9920', 21);

    UPDATE customers
    SET phone_number = '0782 9242 1823'
    WHERE customer_id = 1031;

    DELETE FROM customers
    WHERE customer_id = 1031;
END;
/
```



```
Numarul de clienti dupa inserare: 32
Numarul de clienti neschimbat! A ramas 32
Numarul de clienti dupa stergere: 31


PL/SQL procedure successfully completed.
```

Problema 11. **Cerință:**

Să se realizeze următoarele operații: introduceți, actualizați, ștergeți un joc. Să se anuleze operația dacă se încalcă una din regulile: prețul nu poate fi negativ, seria trebuie să existe deja în baza de date, iar ștergerea unui joc este permisă doar dacă acel joc nu a fost comandat până acum.

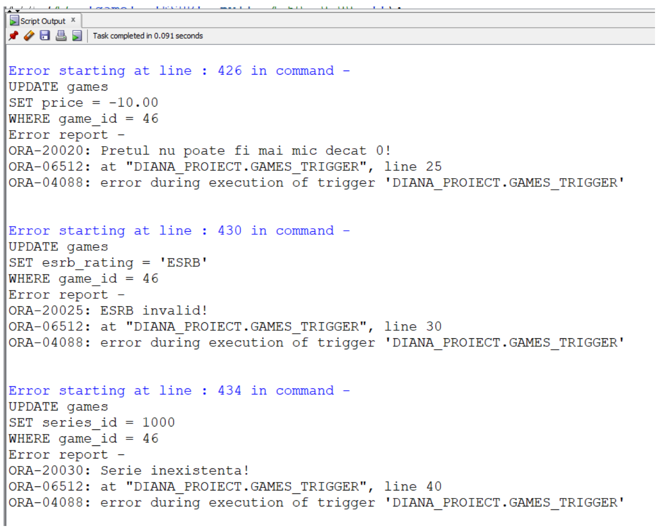
Pentru această problemă, am creat un trigger LMD la nivel de linie, care verifică, pentru fiecare comandă, corectitudinea operației dorite.

```
CREATE OR REPLACE
    TRIGGER games_trigger
BEFORE INSERT OR UPDATE OR DELETE ON games
FOR EACH ROW
DECLARE
    v_nr     NUMBER;
BEGIN

    IF INSERTING THEN
        IF :NEW.price < 0.00 THEN
            RAISE_APPLICATION_ERROR(-20005, 'Pretul nu poate fi mai mic decat 0!');
        END IF;

        IF UPPER(:NEW.esrb_rating) NOT IN ('E','E10+','T','M','AO','RP') THEN
            RAISE_APPLICATION_ERROR(-20010, 'ESRB invalid!');
        END IF;

        SELECT COUNT(*)
        INTO v_nr
        FROM series
        WHERE series_id = :NEW.series_id;

        IF v_nr = 0 AND :NEW.series_id IS NOT NULL THEN
            RAISE_APPLICATION_ERROR(-20015, 'Serie inexistenta!');
        END IF;

    ELSIF UPDATING ('price') THEN
        IF :NEW.price < 0.00 THEN
            RAISE_APPLICATION_ERROR(-20020, 'Pretul nu poate fi mai mic decat 0!');
        END IF;

    ELSIF UPDATING ('esrb_rating') THEN
        IF UPPER(:NEW.esrb_rating) NOT IN ('E','E10+','T','M','AO','RP') THEN
            RAISE_APPLICATION_ERROR(-20025, 'ESRB invalid!');
        END IF;

    ELSIF UPDATING ('series_id') THEN
        SELECT COUNT(*)
        INTO v_nr
        FROM series
        WHERE series_id = :NEW.series_id;

        IF v_nr = 0 OR :NEW.series_id IS NOT NULL THEN
            RAISE_APPLICATION_ERROR(-20030, 'Serie inexistenta!');
        END IF;

    ELSIF DELETING THEN
        SELECT COUNT(*)
        INTO v_nr
        FROM order_game
        WHERE game_id = :OLD.game_id;

        IF v_nr != 0 THEN
            RAISE_APPLICATION_ERROR(-20035, 'Nu este permisa stergerea unui joc care a fost comandat!');
        END IF;
    END IF;
END;
/
```

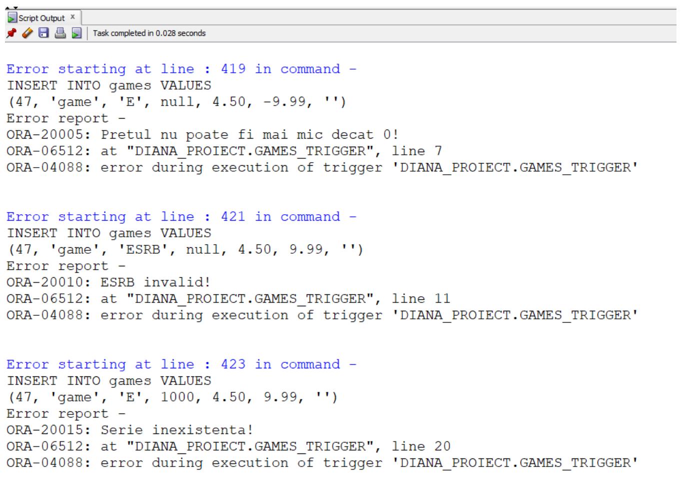
Testarea am realizat-o cu mai multe comenzi SQL, pentru fiecare caz în parte, pentru a declanșa trigger-ul pentru fiecare caz.

```sql
INSERT INTO games VALUES
(47, 'game', 'E', null, 4.50, -9.99, '');
INSERT INTO games VALUES
(47, 'game', 'ESRB', null, 4.50, 9.99, '');
INSERT INTO games VALUES
(47, 'game', 'E', 1000, 4.50, 9.99, '');

UPDATE games
SET price = -10.00
WHERE game_id = 46;

UPDATE games
SET esrb_rating = 'ESRB'
WHERE game_id = 46;

UPDATE games
SET series_id = 1000
WHERE game_id = 46;

DELETE FROM games
WHERE game_id IN (10, 30);
```

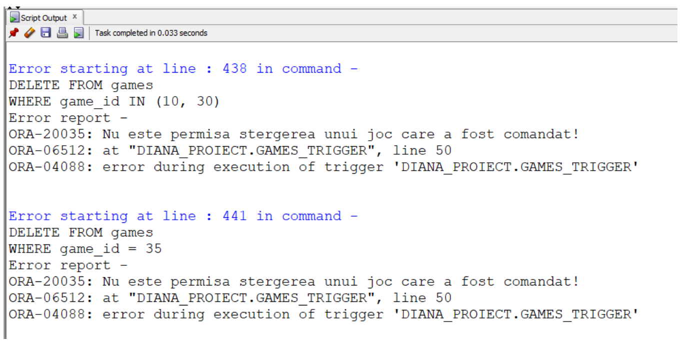
```sql
DELETE FROM games
WHERE game_id = 35;
```



```
Error starting at line : 426 in command -
UPDATE games
SET price = -10.00
WHERE game_id = 46
Error report -
ORA-20020: Pretul nu poate fi mai mic decat 0!
ORA-06512: at "DIANA_PROIECT.GAMES_TRIGGER", line 25
ORA-04088: error during execution of trigger 'DIANA_PROIECT.GAMES_TRIGGER'

Error starting at line : 430 in command -
UPDATE games
SET esrb_rating = 'ESRB'
WHERE game_id = 46
Error report -
ORA-20025: ESRB invalid!
ORA-06512: at "DIANA_PROIECT.GAMES_TRIGGER", line 30
ORA-04088: error during execution of trigger 'DIANA_PROIECT.GAMES_TRIGGER'

Error starting at line : 434 in command -
UPDATE games
SET series_id = 1000
WHERE game_id = 46
Error report -
ORA-20030: Serie inexistenta!
ORA-06512: at "DIANA_PROIECT.GAMES_TRIGGER", line 40
ORA-04088: error during execution of trigger 'DIANA_PROIECT.GAMES_TRIGGER'
```



```
Error starting at line : 419 in command -
INSERT INTO games VALUES
(47, 'game', 'E', null, 4.50, -9.99, '')
Error report -
ORA-20005: Pretul nu poate fi mai mic decat 0!
ORA-06512: at "DIANA_PROIECT.GAMES_TRIGGER", line 7
ORA-04088: error during execution of trigger 'DIANA_PROIECT.GAMES_TRIGGER'

Error starting at line : 421 in command -
INSERT INTO games VALUES
(47, 'game', 'ESRB', null, 4.50, 9.99, '')
Error report -
ORA-20010: ESRB invalid!
ORA-06512: at "DIANA_PROIECT.GAMES_TRIGGER", line 11
ORA-04088: error during execution of trigger 'DIANA_PROIECT.GAMES_TRIGGER'

Error starting at line : 423 in command -
INSERT INTO games VALUES
(47, 'game', 'E', 1000, 4.50, 9.99, '')
Error report -
ORA-20015: Serie inexistenta!
ORA-06512: at "DIANA_PROIECT.GAMES_TRIGGER", line 20
ORA-04088: error during execution of trigger 'DIANA_PROIECT.GAMES_TRIGGER'
```



```
Error starting at line : 438 in command -
DELETE FROM games
WHERE game_id IN (10, 30)
Error report -
ORA-20035: Nu este permisa stergerea unui joc care a fost comandat!
ORA-06512: at "DIANA_PROIECT.GAMES_TRIGGER", line 50
ORA-04088: error during execution of trigger 'DIANA_PROIECT.GAMES_TRIGGER'

Error starting at line : 441 in command -
DELETE FROM games
WHERE game_id = 35
Error report -
ORA-20035: Nu este permisa stergerea unui joc care a fost comandat!
ORA-06512: at "DIANA_PROIECT.GAMES_TRIGGER", line 50
ORA-04088: error during execution of trigger 'DIANA_PROIECT.GAMES_TRIGGER'
```

Problema 12. **Cerință:**

Opriți crearea unui tabel în baza de date în cazul în care numele acestuia începe cu litera 'z' sau dacă este mai lung de 50 de caractere.

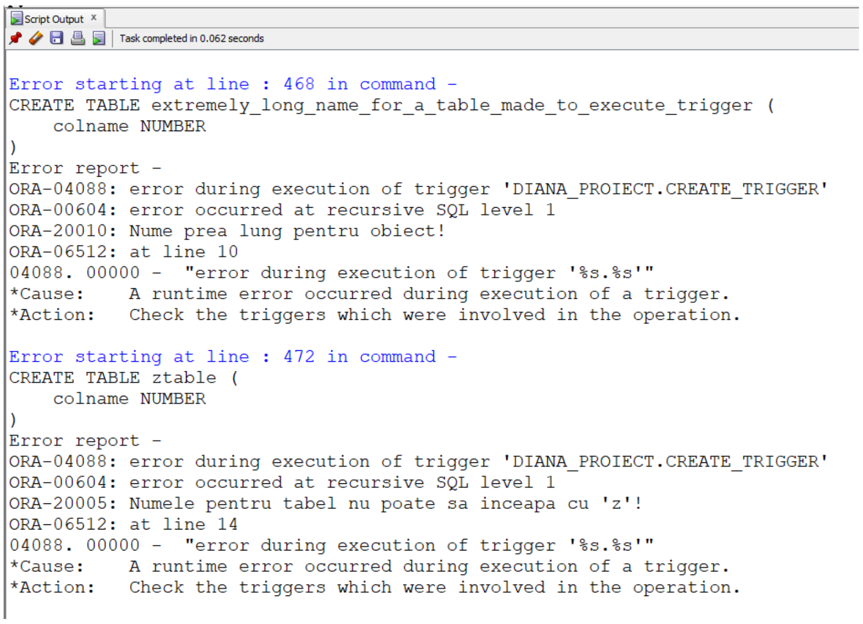
Pentru rezolvarea problemei, am creat un trigger LDD, care se declanșează înainte de crearea unui tabel și verifică cele două condiții pentru numele tabelului.

```
CREATE OR REPLACE
    TRIGGER create_trigger
BEFORE CREATE ON SCHEMA
DECLARE
    v_table_name     user_tables.table_name%TYPE;
BEGIN

    SELECT ora_dict_obj_name
    INTO v_table_name
    FROM DUAL;

    IF LENGTH(v_table_name) > 50 THEN
        RAISE_APPLICATION_ERROR(-20010, 'Nume prea lung pentru obiect!');
    END IF;

    IF UPPER(SUBSTR(v_table_name, 1, 1)) = 'Z' THEN
        RAISE_APPLICATION_ERROR(-20005, 'Numele pentru tabel nu poate sa inceapa cu ''z''!');
    END IF;

END;
/
```

Am încercat crearea a două tabele, pentru a verifica declanșarea trigger-ului.

```
CREATE TABLE extremely_long_name_for_a_table_made_to_execute_trigger (
    colname NUMBER
);
```

```
CREATE TABLE ztable (
    colname NUMBER
);
```



```
Error starting at line : 468 in command -
CREATE TABLE extremely_long_name_for_a_table_made_to_execute_trigger (
    colname NUMBER
)
Error report -
ORA-04088: error during execution of trigger 'DIANA_PROIECT.CREATE_TRIGGER'
ORA-00604: error occurred at recursive SQL level 1
ORA-20010: Nume prea lung pentru obiect!
ORA-06512: at line 10
04088. 00000 -  "error during execution of trigger '%s.%s'"
*Cause:    A runtime error occurred during execution of a trigger.
*Action:   Check the triggers which were involved in the operation.

Error starting at line : 472 in command -
CREATE TABLE ztable (
    colname NUMBER
)
Error report -
ORA-04088: error during execution of trigger 'DIANA_PROIECT.CREATE_TRIGGER'
ORA-00604: error occurred at recursive SQL level 1
ORA-20005: Numele pentru tabel nu poate sa inceapa cu 'z'!
ORA-06512: at line 14
04088. 00000 -  "error during execution of trigger '%s.%s'"
*Cause:    A runtime error occurred during execution of a trigger.
*Action:   Check the triggers which were involved in the operation.
```

Problema 13.

Pentru cerința 13, am creat un pachet numit games_package, care conține toate obiectele necesare rezolvării problemelor 6-9. Corpul tuturor obiectelor din pachet este același ca cele de la cerințele anterioare.

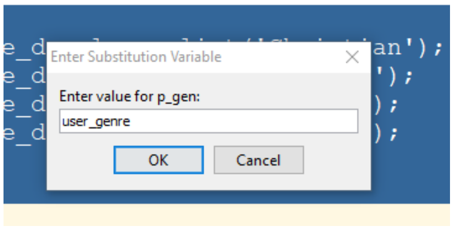
```
-------------------------------------
------ SPECIFICATIA PACHETULUI ------
-------------------------------------

CREATE OR REPLACE PACKAGE games_package
AS

    e_no_games          EXCEPTION;
    e_no_country        EXCEPTION;
    e_no_orders         EXCEPTION;

    TYPE    t_orders        IS TABLE OF     orders.order_id%TYPE;
    TYPE    t_game_id       IS TABLE OF     games.game_id%TYPE;
    TYPE    t_title         IS TABLE OF     games.title%TYPE;
    TYPE    t_developer     IS TABLE OF     developers.developer_name%TYPE;
    TYPE    t_customer_id   IS TABLE OF     customers.customer_id%TYPE;

-- problema 6
    PROCEDURE game_developer_list ( v_genre_name  genres.genre_name%TYPE );

-- problema 7
    PROCEDURE show_modes_game_list;

-- problema 8
    FUNCTION nr_clients_country_orderval ( v_country_name   address.country%TYPE,
                                           v_over           VARCHAR2 )
    RETURN NUMBER;

-- problema 9
    PROCEDURE show_ordered_games_country ( v_country_name     address.country%TYPE );

END games_package;
/

-------------------------------------
--------- CORPUL PACHETULUI ---------
-------------------------------------

CREATE OR REPLACE PACKAGE BODY games_package
AS
-- problema 6
-- Afisati toate titlurile de jocuri video
-- si numele developer-ului fiecaruia care au un gen specificat de catre user.
    PROCEDURE game_developer_list ( v_genre_name  genres.genre_name%TYPE )
    AS
        v_genre_id          game_genre.genre_id%TYPE;
        v_game_id           t_game_id       := t_game_id();
        v_title             t_title         := t_title();
        v_devname           t_developer     := t_developer();
    BEGIN

        dbms_output.put_line('---------- ' || UPPER(v_genre_name) || ' ----------');

        SELECT genre_id
        INTO v_genre_id
        FROM genres
        WHERE genre_name = INITCAP(v_genre_name);
        SELECT game_id
        BULK COLLECT INTO v_game_id
        FROM game_genre
        WHERE genre_id = v_genre_id;

        IF v_game_id.COUNT != 0 THEN
            FOR i IN v_game_id.FIRST .. v_game_id.LAST LOOP

                SELECT title, developer_name
                BULK COLLECT INTO v_title, v_devname
                FROM games
                JOIN game_developer USING (game_id)
                JOIN developers USING (developer_id)
                WHERE game_id = v_game_id(i);

                FOR j IN v_devname.FIRST .. v_devname.LAST LOOP
                    dbms_output.put_line(v_title(j) || '  -  ' || v_devname(j));
                END LOOP;

            END LOOP;
        ELSE
            RAISE e_no_games;
        END IF;

        dbms_output.new_line;

    EXCEPTION
        WHEN e_no_games THEN
            dbms_output.put_line('Nu exista jocuri cu genul dorit!');
            dbms_output.new_line;
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('Gen inexistent!');
            dbms_output.new_line;
        WHEN OTHERS THEN
            dbms_output.put_line('Alta eroare - ' || SQLERRM);
            dbms_output.new_line;
    END game_developer_list;

-- problema 7
-- Pentru fiecare mod, afisati, in ordine alfabetica, numele modului
-- si o lista cu toate jocurile care dispun de acest mod
    PROCEDURE show_modes_game_list
    AS
        v_pos       NUMBER;
        v_number    NUMBER;
    BEGIN

        FOR v_mode IN ( SELECT mode_id, mode_name
                        FROM modes
                        ORDER BY mode_name )
        LOOP
            dbms_output.put_line('--------- ' || UPPER(v_mode.mode_name) || ' ---------');

            SELECT COUNT(*)
            INTO v_number
            FROM game_mode
            WHERE mode_id = v_mode.mode_id;

            dbms_output.put_line('--------- Numar de jocuri: ' || v_number);

            v_pos := 1;
            FOR v_title IN ( SELECT title
                             FROM games
                             JOIN game_mode USING (game_id)
                             WHERE mode_id = v_mode.mode_id )
            LOOP
                dbms_output.put_line(v_pos || '. ' || v_title.title);
                v_pos := v_pos + 1;
            END LOOP;

            dbms_output.new_line;
        END LOOP;
    END show_modes_game_list;
```

```
-- problema 8
-- Calculati numarul total de clienti care traiesc intr-o tara specificata
-- si care au plasat comenzi cu o valoare totala mai mare decat o valoare data
    FUNCTION nr_clients_country_orderval ( v_country_name   address.country%TYPE,
                                           v_over           VARCHAR2 )
    RETURN NUMBER
    AS
        v_customer_id       t_customer_id   := t_customer_id();
        v_order_nr          NUMBER;
        v_number            NUMBER          := 0;
        v_limit             orders.total_price%TYPE;
    BEGIN

        v_limit := TO_NUMBER(v_over);

        SELECT customer_id
        BULK COLLECT INTO v_customer_id
        FROM customers
        WHERE address_id IN ( SELECT address_id
                              FROM address
                              WHERE LOWER(country) = LOWER(v_country_name) );

        IF v_customer_id.COUNT = 0 THEN
            RAISE e_no_country;
        END IF;

        FOR i IN v_customer_id.FIRST .. v_customer_id.LAST LOOP

            SELECT COUNT(DISTINCT customer_id)
            INTO v_order_nr
            FROM orders
            WHERE customer_id = v_customer_id(i)
            AND total_price >= v_limit;

            v_number := v_number + v_order_nr;
        END LOOP;

        RETURN v_number;

    EXCEPTION
        WHEN e_no_country THEN
            dbms_output.put_line('Nu exista clienti cu locuinta in ' || v_country_name || '!');
            RETURN -1;
        WHEN VALUE_ERROR THEN
            dbms_output.put_line('Format gresit pentru valoarea ''' || v_over || '''!');
            RETURN -6502;
        WHEN OTHERS THEN
            dbms_output.put_line('Alta eroare! - ' || SQLERRM || ' - ' || SQLCODE);
            RETURN -20005;

    END nr_clients_country_orderval;

-- problema 9
-- Afisati toate jocurile (titlul lor) care au fost cumparate de clienti
-- care traiesc intr-o tara specificata si castigurile din acea tara.
    PROCEDURE show_ordered_games_country ( v_country_name     address.country%TYPE )
    AS
        v_customer_id           t_customer_id   := t_customer_id();
        v_game_temp             t_game_id       := t_game_id();
        v_game_id               t_game_id       := t_game_id();
        v_order_temp            t_orders        := t_orders();
        v_order_id              t_orders        := t_orders();
        v_price                 t_price         := t_price();
        v_title                 games.title%TYPE;
        v_profit                NUMBER;
    BEGIN
```

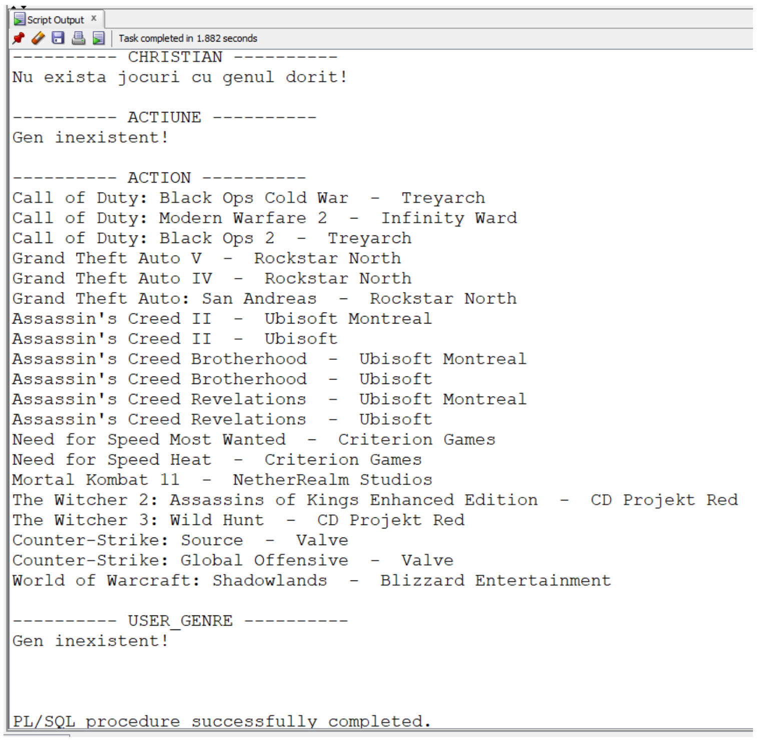
```
        dbms_output.put_line('------- ' || UPPER(v_country_name) || ' -------');

        SELECT customer_id
        BULK COLLECT INTO v_customer_id
        FROM customers
        WHERE address_id IN ( SELECT address_id
                              FROM address
                              WHERE LOWER(country) = LOWER(v_country_name));

        IF v_customer_id.COUNT = 0 THEN
            RAISE e_no_country;
        END IF;

        v_profit := 0;

        FOR i IN v_customer_id.FIRST .. v_customer_id.LAST LOOP
            SELECT order_id, total_price
            BULK COLLECT INTO v_order_temp, v_price
            FROM orders
            WHERE customer_id = v_customer_id(i);

            IF v_order_temp.COUNT != 0 THEN
                FOR j IN v_order_temp.FIRST .. v_order_temp.LAST LOOP
                    v_order_id.EXTEND;
                    v_order_id( v_order_id.COUNT ) := v_order_temp(j);
                    v_profit := v_profit + v_price(j);
                END LOOP;
            END IF;

            SELECT cashback
            BULK COLLECT INTO v_price
            FROM returns
            WHERE customer_id = v_customer_id(i);

            IF v_price.COUNT != 0 THEN
                FOR j IN v_price.FIRST .. v_price.LAST LOOP
                    v_profit := v_profit - v_price(j);
                END LOOP;
            END IF;
        END LOOP;

        IF v_order_id.COUNT = 0 THEN
            RAISE e_no_orders;
        END IF;

        dbms_output.put_line('PROFIT: '  || v_profit);

        FOR i IN v_order_id.FIRST .. v_order_id.LAST LOOP

            SELECT game_id
            BULK COLLECT INTO v_game_temp
            FROM order_game
            WHERE order_id = v_order_id(i);

            FOR j IN v_game_temp.FIRST .. v_game_temp.LAST LOOP
                v_game_id.EXTEND;
                v_game_id( v_game_id.COUNT ) := v_game_temp(j);
            END LOOP;

        END LOOP;

        v_game_id := SET(v_game_id);

        FOR i IN v_game_id.FIRST .. v_game_id.LAST LOOP
            SELECT title
            INTO v_title
            FROM games
            WHERE game_id = v_game_id(i);

            dbms_output.put_line(v_title);
        END LOOP;
        dbms_output.new_line;

    EXCEPTION
        WHEN e_no_country THEN
            dbms_output.put_line('Nu exista clienti care locuiesc in ' || INITCAP(v_country_name)
                                 || '!');
            dbms_output.new_line;
        WHEN e_no_orders THEN
            dbms_output.put_line('Niciun client din ' || INITCAP(v_country_name) ||
                                 ' nu a efectuat comenzi!');
            dbms_output.new_line;
        WHEN OTHERS THEN
            dbms_output.put_line('Alta eroare! - ' || SQLERRM);
            dbms_output.new_line;

    END show_ordered_games_country;

END games_package;
/
```

Verificarea pachetului

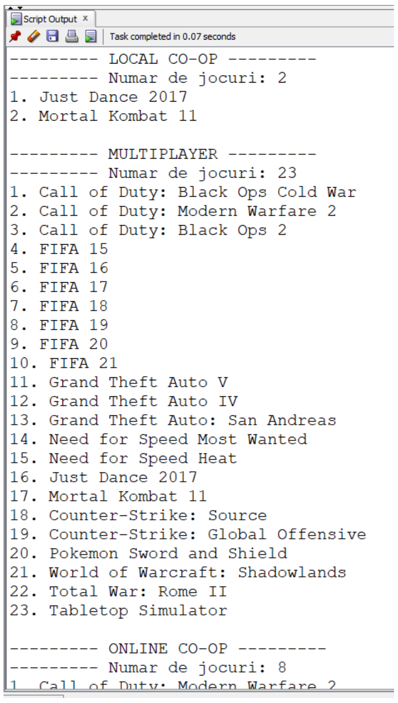
```
-- problema 6
BEGIN
    games_package.game_developer_list('Christian');
    games_package.game_developer_list('Actiune');
    games_package.game_developer_list('Action');
    games_package.game_developer_list('&p_gen');
END;
/
```



Enter value for p_gen:
user_genre



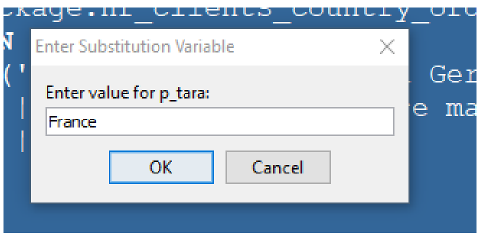
```
---------- CHRISTIAN ----------
Nu exista jocuri cu genul dorit!

---------- ACTIUNE ----------
Gen inexistent!

---------- ACTION ----------
Call of Duty: Black Ops Cold War  -  Treyarch
Call of Duty: Modern Warfare 2  -  Infinity Ward
Call of Duty: Black Ops 2  -  Treyarch
Grand Theft Auto V  -  Rockstar North
Grand Theft Auto IV  -  Rockstar North
Grand Theft Auto: San Andreas  -  Rockstar North
Assassin's Creed II  -  Ubisoft Montreal
Assassin's Creed II  -  Ubisoft
Assassin's Creed Brotherhood  -  Ubisoft Montreal
Assassin's Creed Brotherhood  -  Ubisoft
Assassin's Creed Revelations  -  Ubisoft Montreal
Assassin's Creed Revelations  -  Ubisoft
Need for Speed Most Wanted  -  Criterion Games
Need for Speed Heat  -  Criterion Games
Mortal Kombat 11  -  NetherRealm Studios
The Witcher 2: Assassins of Kings Enhanced Edition  -  CD Projekt Red
The Witcher 3: Wild Hunt  -  CD Projekt Red
Counter-Strike: Source  -  Valve
Counter-Strike: Global Offensive  -  Valve
World of Warcraft: Shadowlands  -  Blizzard Entertainment

---------- USER_GENRE ----------
Gen inexistent!


PL/SQL procedure successfully completed.
```



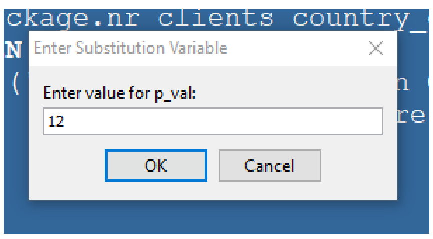
```
--------- LOCAL CO-OP ---------
--------- Numar de jocuri: 2
1. Just Dance 2017
2. Mortal Kombat 11

--------- MULTIPLAYER ---------
--------- Numar de jocuri: 23
1. Call of Duty: Black Ops Cold War
2. Call of Duty: Modern Warfare 2
3. Call of Duty: Black Ops 2
4. FIFA 15
5. FIFA 16
6. FIFA 17
7. FIFA 18
8. FIFA 19
9. FIFA 20
10. FIFA 21
11. Grand Theft Auto V
12. Grand Theft Auto IV
13. Grand Theft Auto: San Andreas
14. Need for Speed Most Wanted
15. Need for Speed Heat
16. Just Dance 2017
17. Mortal Kombat 11
18. Counter-Strike: Source
19. Counter-Strike: Global Offensive
20. Pokemon Sword and Shield
21. World of Warcraft: Shadowlands
22. Total War: Rome II
23. Tabletop Simulator

--------- ONLINE CO-OP ---------
--------- Numar de jocuri: 8
1. Call of Duty: Modern Warfare 2
```

```
-- problema 7
BEGIN
    games_package.show_modes_game_list;

END;
/
```

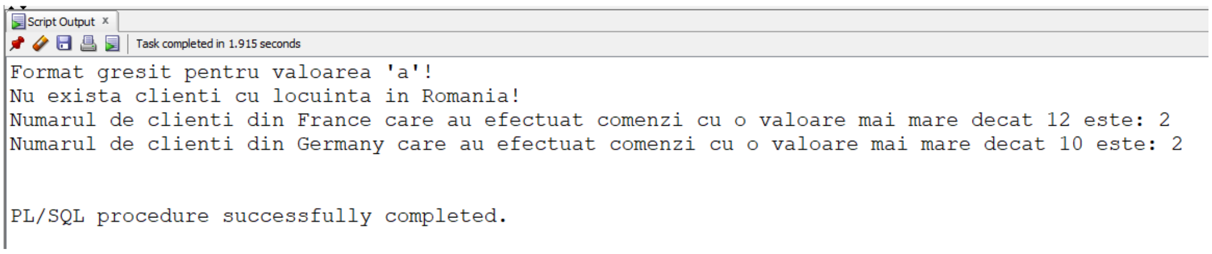
```
-- problema 8
DECLARE
    v_return_val    NUMBER;
    v_tara          VARCHAR2(200)  := '&p_tara';
    v_val           VARCHAR2(200)  := '&p_val';
BEGIN
    -- test 1
    v_return_val := games_package.nr_clients_country_orderval ('Germany', 'a');
    IF v_return_val >= 0 THEN
        dbms_output.put_line('Numarul de clienti din Germany care au efectuat '
                             || 'comenzi cu o valoare mai mare decat 10 este: '
                             || v_return_val);
    END IF;

    -- test 2
    v_return_val := games_package.nr_clients_country_orderval ('Romania', 100);
    IF v_return_val >= 0 THEN
        dbms_output.put_line('Numarul de clienti din Romania care au efectuat '
                             || 'comenzi cu o valoare mai mare decat 100 este: '
                             || v_return_val);
    END IF;

    -- test 3
    v_return_val := games_package.nr_clients_country_orderval (v_tara, v_val);
    IF v_return_val >= 0 THEN
        dbms_output.put_line('Numarul de clienti din ' || INITCAP(v_tara) || ' care au efectuat '
                             || 'comenzi cu o valoare mai mare decat ' || v_val || ' este: '
                             || v_return_val);
    END IF;

    -- test 4
    v_return_val := games_package.nr_clients_country_orderval ('Germany', 10);
    IF v_return_val >= 0 THEN
        dbms_output.put_line('Numarul de clienti din Germany care au efectuat '
                             || 'comenzi cu o valoare mai mare decat 10 este: '
                             || v_return_val);
    END IF;
END;
/
```



Enter value for p_tara: France

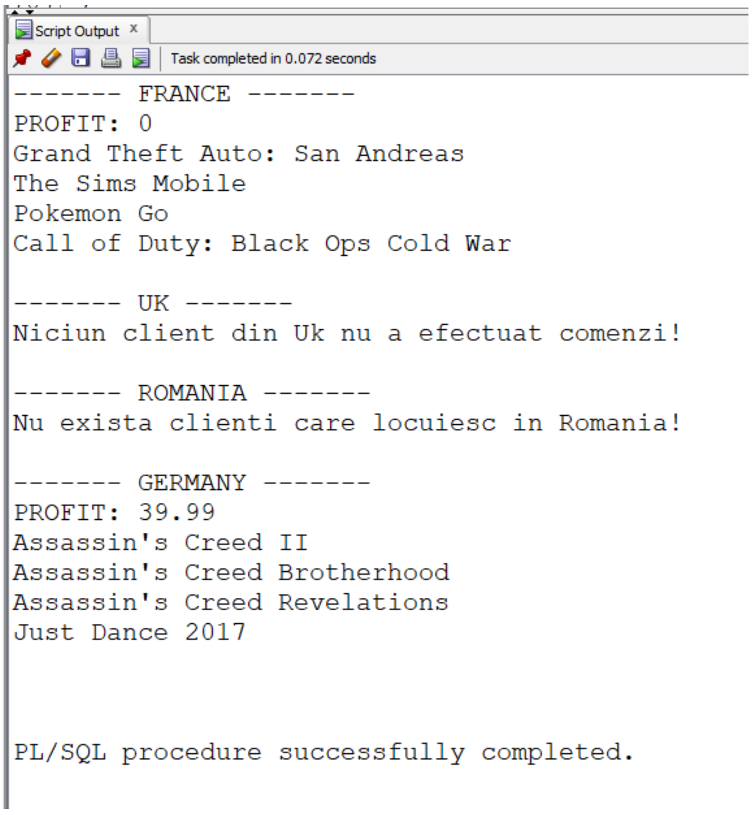


Enter value for p_val: 12



```
Format gresit pentru valoarea 'a'!
Nu exista clienti cu locuinta in Romania!
Numarul de clienti din France care au efectuat comenzi cu o valoare mai mare decat 12 este: 2
Numarul de clienti din Germany care au efectuat comenzi cu o valoare mai mare decat 10 este: 2


PL/SQL procedure successfully completed.
```

```
-- problema 9
BEGIN
    games_package.show_ordered_games_country('France');
    games_package.show_ordered_games_country('UK');
    games_package.show_ordered_games_country('Romania');
    games_package.show_ordered_games_country('Germany');
END;
```



```
------- FRANCE -------
PROFIT: 0
Grand Theft Auto: San Andreas
The Sims Mobile
Pokemon Go
Call of Duty: Black Ops Cold War

------- UK -------
Niciun client din Uk nu a efectuat comenzi!

------- ROMANIA -------
Nu exista clienti care locuiesc in Romania!

------- GERMANY -------
PROFIT: 39.99
Assassin's Creed II
Assassin's Creed Brotherhood
Assassin's Creed Revelations
Just Dance 2017



PL/SQL procedure successfully completed.
```

Problema 14.

Pentru problema 14, am creat un pachet numit extra_package, care conține mai multe obiecte pentru a îndeplini următoarele cerințe:

1. schimbarea, cu un procent dat, a prețurilor tuturor jocurilor care au fost publicate de un anumit publisher
2. calcularea prețului total pentru jocurile publicate de un anumit publisher
3. notificarea clienților cu un text customizat
4. afișarea top 3 cele mai ieftine jocuri publicate de un anumit publisher

Pentru această problemă, a fost necesară crearea unei secvențe.

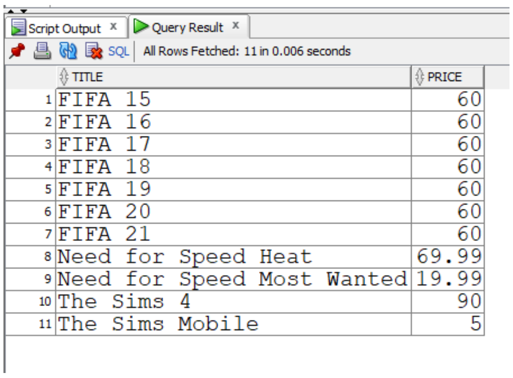
```
CREATE SEQUENCE notification_seq
INCREMENT BY 1
START WITH 6;
```

```
-------------------------------------
------ SPECIFICATIA PACHETULUI ------
-------------------------------------
CREATE OR REPLACE
    PACKAGE extra_package
AS

    TYPE    t_games     IS TABLE OF         games.game_id%TYPE INDEX BY BINARY_INTEGER;
    TYPE    arr_games   IS VARRAY(3) OF     games.game_id%TYPE;

    FUNCTION get_publisher_id ( v_publisher_name   publishers.publisher_name%TYPE )
    RETURN publishers.publisher_id%TYPE;

    PROCEDURE change_price ( v_publisher_name   publishers.publisher_name%TYPE,
                             v_percentage       NUMBER );

    FUNCTION calculate_price_per_publisher ( v_publisher_name   publishers.publisher_name%TYPE )
    RETURN NUMBER;

    PROCEDURE notify_clients ( v_text    notifications.text%TYPE );

    PROCEDURE show_top3_cheapest_games ( v_publisher_name   publishers.publisher_name%TYPE );

    CURSOR c_customer_id IS
        SELECT customer_id
        FROM customers
        ORDER BY customer_id;

    CURSOR c_top3_game_id ( v_publisher_name   publishers.publisher_name%TYPE ) IS
        SELECT game_id
        FROM ( SELECT DISTINCT game_id, price
               FROM games
               JOIN game_publisher_platform USING (game_id)
               JOIN publishers USING (publisher_id)
               WHERE LOWER(publisher_name) = 'ea'
               ORDER BY price )
        WHERE ROWNUM <= 3;

    e_no_games          EXCEPTION;
    e_wrong_percentage  EXCEPTION;

END extra_package;
/
```

```
-------------------------------------
--------- CORPUL PACHETULUI ---------
-------------------------------------
CREATE OR REPLACE PACKAGE BODY extra_package
AS
    FUNCTION get_publisher_id ( v_publisher_name   publishers.publisher_name%TYPE )
    RETURN publishers.publisher_id%TYPE
    AS
        v_id            publishers.publisher_id%TYPE;
    BEGIN

        SELECT publisher_id
        INTO v_id
        FROM publishers
        WHERE LOWER(publisher_name) = LOWER(v_publisher_name);

        RETURN v_id;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('Nu exista publisher-ul cerut!');
            RETURN -20001;
        WHEN OTHERS THEN
            dbms_output.put_line('Alta eroare! - ' || SQLERRM);
            RETURN -20000;
    END get_publisher_id;

    FUNCTION calculate_price_per_publisher ( v_publisher_name   publishers.publisher_name%TYPE )
    RETURN NUMBER
    AS
        v_publisher_id      publishers.publisher_id%TYPE;
        v_price             NUMBER;
    BEGIN
        v_publisher_id := extra_package.get_publisher_id(v_publisher_name);

        SELECT SUM(price)
        INTO v_price
        FROM games
        JOIN game_publisher_platform USING (game_id)
        WHERE publisher_id = v_publisher_id;

        RETURN v_price;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('Publisher fara jocuri publicate!');
            RETURN -1;
        WHEN OTHERS THEN
            dbms_output.put_line('Alta eroare! - ' || SQLERRM);
            RETURN -20005;
    END calculate_price_per_publisher;

    PROCEDURE change_price ( v_publisher_name   publishers.publisher_name%TYPE,
                             v_percentage       NUMBER )
    AS
        v_game_id           t_games;
        v_publisher_id      publishers.publisher_id%TYPE;
        v_old_sum           NUMBER;
        v_new_sum           NUMBER;
        v_notif_text        notifications.text%TYPE;
    BEGIN

        IF v_percentage < -1 OR v_percentage > 1 OR v_percentage = 0 THEN
            RAISE e_wrong_percentage;
        END IF;

        v_publisher_id := extra_package.get_publisher_id(v_publisher_name);

        SELECT DISTINCT game_id
        BULK COLLECT INTO v_game_id
        FROM game_publisher_platform
        WHERE publisher_id = v_publisher_id;
        IF v_game_id.COUNT != 0 THEN

            v_old_sum := extra_package.calculate_price_per_publisher(v_publisher_name);

            FOR i IN v_game_id.FIRST .. v_game_id.LAST LOOP

                UPDATE games
                SET price = price + v_percentage * price
                WHERE game_id = v_game_id(i);

            END LOOP;

            v_new_sum := extra_package.calculate_price_per_publisher(v_publisher_name);

            IF v_new_sum < v_old_sum THEN
                v_notif_text := 'Publisher ' || UPPER(v_publisher_name) ||
                                ' has changed prices of all their games! ' ||
                                'If you buy all the games you would save $' ||
                                (v_old_sum - v_new_sum) || '!';
            ELSE
                v_notif_text := 'Publisher ' || UPPER(v_publisher_name) ||
                                ' has changed prices of all their games! ' ||
                                'Guess the sale is over...';
            END IF;

            extra_package.notify_clients(v_notif_text);
        ELSE
            RAISE e_no_games;
        END IF;

    EXCEPTION
        WHEN e_no_games THEN
            dbms_output.put_line('Nu exista jocuri cu publisher-ul dorit!');
        WHEN e_wrong_percentage THEN
            dbms_output.put_line('Procent invalid!');
        WHEN OTHERS THEN
            dbms_output.put_line('Alta eroare! - ' || SQLERRM);
    END change_price;

    PROCEDURE notify_clients ( v_text    notifications.text%TYPE )
    AS
    BEGIN
        FOR customer IN extra_package.c_customer_id LOOP

            INSERT INTO notifications VALUES
            ( notification_seq.NEXTVAL,
              customer.customer_id,
              v_text,
              TO_CHAR(SYSDATE, 'DD-MON-YY'),
              0
            );

        END LOOP;
    END notify_clients;

    PROCEDURE show_top3_cheapest_games ( v_publisher_name   publishers.publisher_name%TYPE )
    AS
        v_games             arr_games := arr_games();
        v_publisher_id      publishers.publisher_id%TYPE;
        v_game_allinfo      games%ROWTYPE;
        v_temp              games.game_id%TYPE;
        v_number            NUMBER;
    BEGIN
        v_publisher_id := extra_package.get_publisher_id(v_publisher_name);

        OPEN extra_package.c_top3_game_id(v_publisher_name);

            LOOP
                FETCH extra_package.c_top3_game_id INTO v_temp;
                EXIT WHEN extra_package.c_top3_game_id%NOTFOUND;
                v_games.EXTEND;
                v_games(v_games.COUNT) := v_temp;
            END LOOP;
        CLOSE extra_package.c_top3_game_id;

        dbms_output.put_line('------- TOP 3 CELE MAI IEFTINE -------');
        dbms_output.put_line('------- ' || UPPER(v_publisher_name) || ' -------');

        FOR i IN 1..3 LOOP

            SELECT *
            INTO v_game_allinfo
            FROM games
            WHERE game_id = v_games(i);

            SELECT COUNT(DISTINCT game_id)
            INTO v_number
            FROM order_game
            WHERE game_id = v_games(i);

            dbms_output.put_line('****************************************');
            dbms_output.put_line(v_game_allinfo.title);
            dbms_output.put_line(' -->  PRET: $' || v_game_allinfo.price);
            dbms_output.put_line(' --> NOTA: ' || v_game_allinfo.review);
            dbms_output.put_line(' --> NR. COMENZI: ' || v_number);
            dbms_output.new_line;

        END LOOP;

    END show_top3_cheapest_games;

END extra_package;
/
```

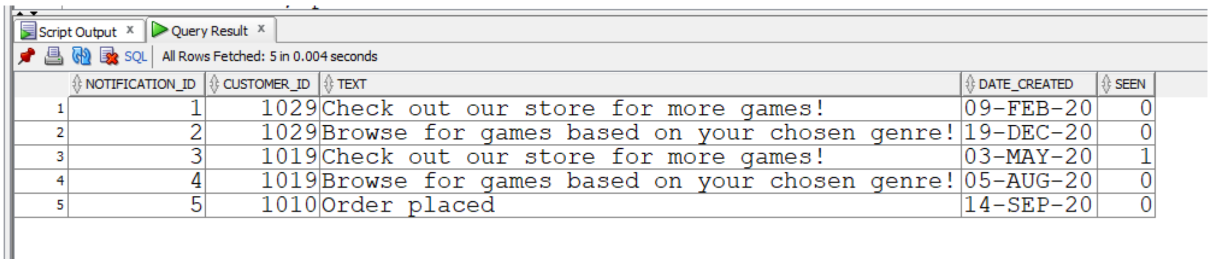
Verificarea pachetului

```
SELECT title, price
FROM games
JOIN game_publisher_platform USING (game_id)
JOIN publishers USING (publisher_id)
WHERE publisher_id = 2
GROUP BY title, price
ORDER BY title;
```


Script Output × | Query Result ×
SQL | All Rows Fetched: 11 in 0.006 seconds

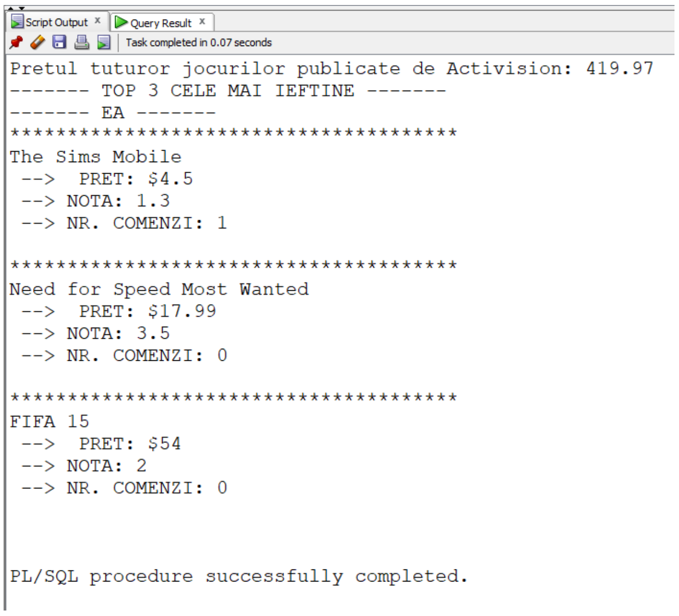
	TITLE	PRICE
1	FIFA 15	60
2	FIFA 16	60
3	FIFA 17	60
4	FIFA 18	60
5	FIFA 19	60
6	FIFA 20	60
7	FIFA 21	60
8	Need for Speed Heat	69.99
9	Need for Speed Most Wanted	19.99
10	The Sims 4	90
11	The Sims Mobile	5

```
SELECT *
FROM notifications;
```


Script Output × | Query Result ×
SQL | All Rows Fetched: 5 in 0.004 seconds

	NOTIFICATION_ID	CUSTOMER_ID	TEXT	DATE_CREATED	SEEN
1	1	1029	Check out our store for more games!	09-FEB-20	0
2	2	1029	Browse for games based on your chosen genre!	19-DEC-20	0
3	3	1019	Check out our store for more games!	03-MAY-20	1
4	4	1019	Browse for games based on your chosen genre!	05-AUG-20	0
5	5	1010	Order placed	14-SEP-20	0

```
BEGIN
    dbms_output.put_line('Pretul tuturor jocurilor publicate de Activision: ' ||
                         extra_package.calculate_price_per_publisher('Activision'));

    extra_package.change_price('EA', -0.1);

    extra_package.show_top3_cheapest_games('EA');
END;
/
```

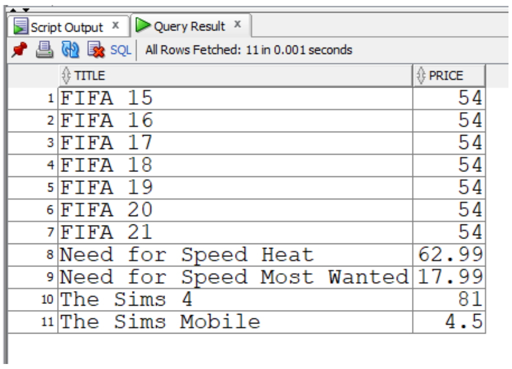


Script Output × | Query Result ×

Task completed in 0.07 seconds

```
Pretul tuturor jocurilor publicate de Activision: 419.97
------- TOP 3 CELE MAI IEFTINE -------
------- EA -------
***************************************
The Sims Mobile
 -->  PRET: $4.5
 --> NOTA: 1.3
 --> NR. COMENZI: 1

***************************************
Need for Speed Most Wanted
 -->  PRET: $17.99
 --> NOTA: 3.5
 --> NR. COMENZI: 0

***************************************
FIFA 15
 -->  PRET: $54
 --> NOTA: 2
 --> NR. COMENZI: 0



PL/SQL procedure successfully completed.
```

```
SELECT title, price
FROM games
JOIN game_publisher_platform USING (game_id)
JOIN publishers USING (publisher_id)
WHERE publisher_id = 2
GROUP BY title, price
ORDER BY title;
```

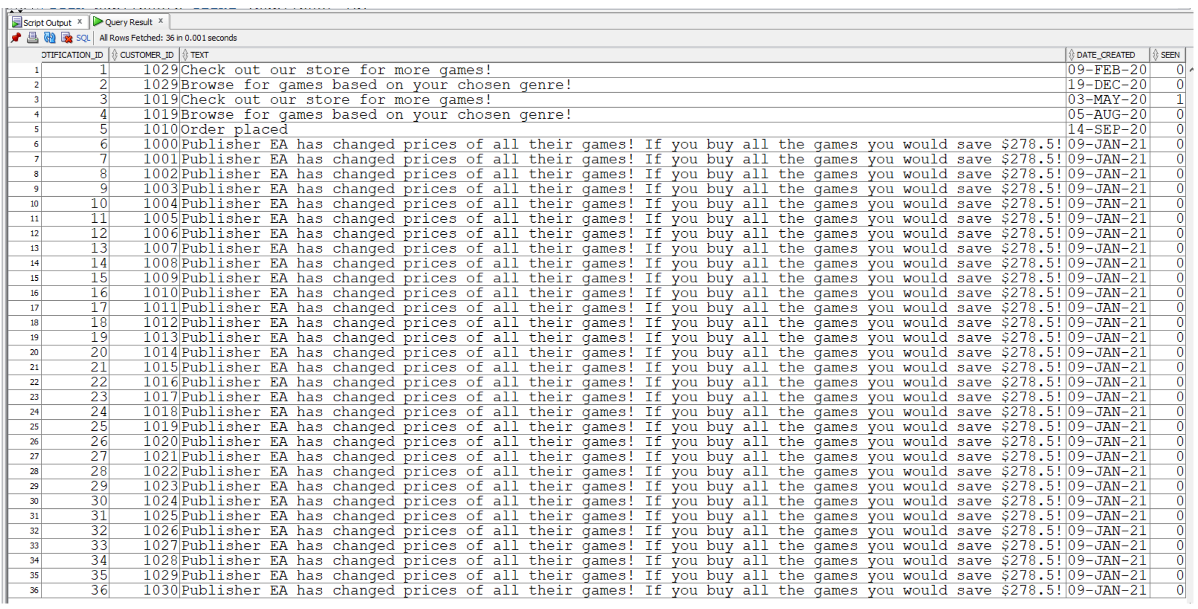


Script Output × | Query Result ×

SQL | All Rows Fetched: 11 in 0.001 seconds

	TITLE	PRICE
1	FIFA 15	54
2	FIFA 16	54
3	FIFA 17	54
4	FIFA 18	54
5	FIFA 19	54
6	FIFA 20	54
7	FIFA 21	54
8	Need for Speed Heat	62.99
9	Need for Speed Most Wanted	17.99
10	The Sims 4	81
11	The Sims Mobile	4.5

```
SELECT *
FROM notifications;
```



Script Output × | Query Result ×

SQL | All Rows Fetched: 36 in 0.001 seconds

	OTIFICATION_ID	CUSTOMER_ID	TEXT	DATE_CREATED	SEEN
1	1	1029	Check out our store for more games!	09-FEB-20	0
2	2	1029	Browse for games based on your chosen genre!	19-DEC-20	0
3	3	1019	Check out our store for more games!	03-MAY-20	1
4	4	1019	Browse for games based on your chosen genre!	05-AUG-20	0
5	5	1010	Order placed	14-SEP-20	0
6	6	1000	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
7	7	1001	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
8	8	1002	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
9	9	1003	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
10	10	1004	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
11	11	1005	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
12	12	1006	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
13	13	1007	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
14	14	1008	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
15	15	1009	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
16	16	1010	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
17	17	1011	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
18	18	1012	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
19	19	1013	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
20	20	1014	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
21	21	1015	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
22	22	1016	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
23	23	1017	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
24	24	1018	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
25	25	1019	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
26	26	1020	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
27	27	1021	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
28	28	1022	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
29	29	1023	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
30	30	1024	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
31	31	1025	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
32	32	1026	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
33	33	1027	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
34	34	1028	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
35	35	1029	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0
36	36	1030	Publisher EA has changed prices of all their games! If you buy all the games you would save $278.5!	09-JAN-21	0