# PROIECT SISTEME DE GESTIONARE A BAZELOR DE DATE

## Mai jos veți găsi exercițiile de la 1 la 5 inclusiv

În acest proiect voi prezenta organizarea concertelor cu ajutorul bazelor de date. Această organizare este esențială din punct de vedere al gestionării tuturor rolurilor ce trebuie îndeplinite pentru susținerea concertelor, rezolvând eventualele probleme ce ar putea apărea pe parcursul organizării. De asemenea, bazele de date conferă claritate fiecărei atribute prezente in tabel. Tabela CONCERT reprezintă nucleul diagramei conceptuale. În organizarea acestuia trebuie să avem în vedere locul în care se va desfășura evenimentul. Astfel apare tabela VENUE ce este legată de CONCERT prin VENUE_PRICE ce va reprezentă prețul închirierii locului respectiv pentru concerte diferite și pentru perioade diferite. În plus, organizarea concertului nu poate fi facută fără ajutorul angajaților. Așa apare tabela EMPLOYEE legată de CONCERT prin tabela de legătură EMPLOYEE_CONCERT. EMPLOYEE_CONCERT va gestiona rolul fiecărui angajat și grupul din care face parte prin tabelele ROLE și GRUP. Tabela TICKET este legată de CONCERT prin tabela de legătură TICKET_CONCERT. TICKET reprezintă biletele cumparate pentru fiecare concert in parte. Totodată, un asemenea eveniment nu poate fi organizat fără un moment artistic muzical, deci este necesară tabela ARTIST legată de CONCERT prin tabela de legătură CONCERT_ARTIST_PRICE ce va reprezenta pretul artistului pentru fiecare concert în parte, iar după cum știm un artist are nevoie de un impresar bun pentru a susține cat mai multe recitaluri, tabela răspunzătoare pentru gestionarea impresarilor fiind MANAGER. Un element crucial ce ține de organizarea concertului este sponsorizarea, sponsorii aparținând tabelei SPONSOR, tabelă legată de CONCERT prin SPONSORSHIP deoarece putem avea același sponsor la mai multe evenimente.

-Tabela CONCERT va conține informații despre fiecare concert în parte.

-Este necesar un număr mare de bilete, astfel se va constitui numărul de participanți la concert (ce trebuie să fie cât mai mare), iar banii strânși din vânzarea biletelor vor fi intra în alcătuirea bugetului prevăzut de organizarea evenimentului.

-Numărul de angajați trebuie să fie de asemenea ridicat, iar fiecare angajat va face parte dintr-un grup și va avea un rol.

-La un concert poate cânta un singur artist, iar acesta va avea un preț diferit în funcție de evenimentul la care este chemat.

-Un artist poate avea un singur manager care necesită 5 ani de experiență în acest domeniu.

-Un concert poate fi într-un singur loc, iar locația va avea un preț în funcție de concert și perioadă.

-Concertul poate avea mai mulți sponsori, iar fiecare sponsor va contribui cu o sumă.

-Un angajat poate participa la mai multe concerte.

-Un sponsor poate sponsoriza mai multe concerte.

-Un artist poate cânta la mai multe concerte.

-Un manager poate lucra cu mai mulți artiști.

-Se pot organiza mai multe concerte într-o locație.

Pentru modelul de date referitor la organizarea concertelor, structurile CONCERT, TICKET, VENUE_PRICE, VENUE, SPONSOR, CONCERT_ARTIST_PRICE, ARTIST, MANAGER, EMPLOYEE, EMPLOYEE_CONCERT, ROLE și GRUP reprezintă entități. Vom prezenta entităţile modelului de date, dând o descriere completă a fiecăreia. De asemenea, pentru fiecare entitate se va preciza cheia primară. Toate entităţile care vor fi prezentate sunt independente, cu excepţia entităţilor dependente ROLE și GROUP.

-CONCERT = este tabela principală, în jurul căreia se orientează toți factori spre organizarea unui concert. Aceasta conține datele principale, cum ar fi numele evenimentului și ziua în care are loc. Cheia primară a entității este ID.

-TICKET = reprezintă biletul cumpărat de fiecare persoană în parte. Acesta va avea un preț diferit în funcție de concertul care are loc și în funcție de tipul persoanei ce îl deține (copil, student, adult). Cheia primară a entității este ID.

-VENUE_PRICE = entitate dependentă de VENUE. Este o tabelă ce va afișa prețul închirierii unei locații in funcție de concert și în funcție de perioada în care are loc. Cheia primară este compusă aici din CONCERT_ID și VENUE_ID.

-VENUE = entitate ce conține informații utile despre locurile disponibile în vederea susținerii unui concert (adresa, oraș, etc.). Cheia primară a entității este ID.

-SPONSOR = persoană fizică sau juridică ce contribuie financiar sau în altă manieră la desfăşurarea prezentării de modă. Cheia primară a acestei entităţi este ID.

-CONCERT_ARTIST_PRICE = entitate dependentă de ARTIST ce va deține suma cerută de un fiecare artist în parte pentru a susține un concert. Această sumă va fi diferi de la concert la concert. Cheia primară a entității este compusa din CONCERT_ID și ARTIST_ID.

-ARTIST = persoană fizică sau juridică selectată pentru a-și susține momentul artistic în cadrul unui concert. Cheia primară a entității este ID.

-MANAGER = persoană fizică sau juridică răspunzătoare pentru încasările artistului cu care este în colaborare. Cheia primară a entității este ID.

-EMPLOYEE = persoană fizică sau juridică ce ajută la organizarea unuia sau a mai multor concerte. Cheia primară a entității este ID.

-EMPLOYEE_CONCERT = identifică fiecare angajat în parte și îl repartizează în funcție de rol si de grupul din care face parte. Cheia primară este compusă din CONCERT_ID și EMPLOYEE_ID.

-ROLE = entitate dependentă de EMPLOYEE_CONCERT ce constituie rolul fiecărui angajat în parte. Cheia primară a entității este ID.

-GRUP = entitate dependentă de EMPLOYEE_CONCERT ce constituie gupul din care fiecare angajat face parte. Cheia primară a entității este ID.

Voi prezenta relaţiile modelului de date, dând o descriere completă a fiecăreia. De fapt, denumirile acestor legături sunt sugestive, reflectând conţinutul acestora şi entităţile pe care le leagă. Pentru fiecare relaţie se va preciza cardinalitatea minimă şi maximă.

-CONCERT_are_TICKET = relaţie care leagă entităţile CONCERT şi TICKET, reflectând legătura dintre acestea (ce bilete are un concert). Relaţia are cardinalitatea minimă 1:0 şi cardinalitatea maximă m:n.

-CONCERT_are_VENUE = relaţie care leagă entităţile CONCERT şi VENUE, reflectând legătura dintre acestea (ce locație are un concert). Relaţia are cardinalitatea minimă 1:0 şi cardinalitatea maximă m:n.

-VENUE_PRICE_are_CONCERT = relaţie care leagă entităţile VENUE_PRICE şi CONCERT, reflectând legătura dintre acestea (care este prețul pentru concert). Relaţia are cardinalitatea minimă 1:1 şi cardinalitatea maximă 1:n.

-VENUE_are_VENUE_PRICE = relaţie care leagă entităţile VENUE şi VENUE_PRICE, reflectând legătura dintre acestea (ce preț are locul respectiv). Relaţia are cardinalitatea minimă 1:1 şi cardinalitatea maximă m:n.

-SPONSOR_finanțează_CONCERT = relaţie care leagă entităţile SPONSOR şi CONCERT, reflectând legătura dintre acestea (ce concert este finanțat de sponsor). Relaţia are cardinalitatea minimă 0:1 şi cardinalitatea maximă m:n.

-ARTIST_cântă_la_CONCERT = relaţie care leagă entităţile ARTIST şi CONCERT, reflectând legătura dintre acestea (care artist cântă la concert). Relaţia are cardinalitatea minimă 0:1 şi cardinalitatea maximă n:1.

-ARTIST_are_CONCERT_ARTIST_PRICE = relaţie care leagă entităţile ARTIST şi CONCERT_ARTIST_PRICE, reflectând legătura dintre acestea (ce sumă cere un artist). Relaţia are cardinalitatea minimă 1:1 şi cardinalitatea maximă m:n.

-CONCERT_ARTIST_PRICE_are_CONCERT = relaţie care leagă entităţile CONCERT_ARTIST_PRICE şi CONCERT, reflectând legătura dintre acestea (care este preţul cerut de artist pentru un anumit concert). Relaţia are cardinalitatea minimă 1:1 şi cardinalitatea maximă n:1.

-ARTIST_are_MANAGER = relaţie care leagă entităţile ARTIST şi MANAGER, reflectând legătura dintre acestea (ce manager are un artist). Relaţia are cardinalitatea minimă 1:0 şi cardinalitatea maximă 1:n.

-CONCERT_are_EMPLOYEE = relaţie care leagă entităţile CONCERT şi, reflectând legătura dintre acestea (ce angajați lucrează la organizarea unui concert). Relaţia are cardinalitatea minimă 1:0 şi cardinalitatea maximă m:n.

-EMPLOYEE_are_ROLE = relaţie care leagă entităţile EMPLOYEE şi ROLE, reflectând legătura dintre acestea (care sunt rolurile pe care le poate avea un angajat). Relaţia are cardinalitatea minimă 0:1 şi cardinalitatea maximă 1:n.

-EMPLOYEE_are_GRUP = relaţie care leagă entităţile EMPLOYEE şi GRUP, reflectând legătura dintre acestea (care sunt grupurile din care angajații fac parte). Relaţia are cardinalitatea minimă 0:1 şi cardinalitatea maximă m:n.

Toate cheile primare și toate cheile străine vor fi diferite de null. Mai jos găsiți descrierea atributelor:

a)CONCERT:

CONCERT_BUDGET = variabilă de tip întreg, de lungime maximă 10, care reprezintă bugetul concertului (concert_budget>10000).

EVENT_NAME = variabilă de tip caracter, de lungime maxima 20, care reprezintă numele evenimentului (event_name is not null).

DATE = variabilă de tip data calendaristică, care reprezintă data în care concertul va avea loc.

b)TICKET_CONCERT:

TICKET_ID = variabilă de tip întreg, de lungime maximă 10, cheie primară, reprezintă id-ul unui bilet.

CONCERT_ID = variabilă de tip întreg, de lungime maximă 10, cheie primară, reprezintă id-ul unui concert.

c)TICKET

TYPE = variabilă de tip caracter, de lungime maximă 20, reprezintă tipul biletului (type in ('adult','child','student')).

PRICE = variabilă de tip întreg, de lungime maxima 10, reprezintă prețul unui bilet (price>=10).

d) RENT_PRICE

CONCERT_ID = variabilă de tip întreg, de lungime maximă 10, cheie primară, reprezintă id-ul unui concert.

VENUE_ID = variabilă de tip întreg, de lungime maximă 10, cheie primară, reprezintă id-ul unei locații.

PRICE = variabilă de tip întreg, de lungime maximă 10, reprezintă prețul unei locații pentru un concert (price>=200).

e) VENUE

NAME = variabilă de tip caracter, de lungime maximă 20, reprezintă numele unei locații (name is not null).

COUNTRY = variabilă de tip caracter, de lungime maximă 20, reprezintă numele țării în care se află locația.

CITY = variabilă de tip caracter, de lungime maximă 20, reprezintă numele numele orașului în care se află locația.

ADDRESS = variabilă de tip caracter, de lungime maximă 20, reprezintă adresa locației (address is not null).

f) SPONSORSHIP

CONCERT_ID = variabilă de tip întreg, de lungime maximă 10, cheie primară, reprezintă id-ul unui concert.

SPONSOR_ID = variabilă de tip întreg, de lungime maximă 10, cheie primară, reprezintă id-ul unui sponsor.

g) SPONSOR

NAME = variabilă de tip character, de lungime maximă 20, reprezintă numele unui sponsor (name is not null).

INVESTMENT = variabilă de tip întreg, de lungime maximă 10, reprezintă investiția făcută de un sponsor (investment>=100).

h) CONCERT_ARTIST_PRICE

CONCERT_ID = variabilă de tip întreg, de lungime maximă 10, cheie primară, reprezintă id-ul unui concert.

ARTIST_ID = variabilă de tip întreg, de lungime maximă 10, cheie primară, reprezintă id-ul unui artist.

PRICE = variabilă de tip întreg, de lungime maximă 10, ce reprezintă prețul cerut de artist pentru un concert (price>=500).

i)ARTIST

MANAGER_ID = variabilă de tip întreg, de lungime maximă 10, cheie străină, reprezintă id-ul unui manager.

NAME = variabilă de tip caracter, de lungime maximă 20, reprezintă numele unui artist (name is not null).

MUSIC_GENRE = variabilă de tip caracter, de lungime maximă 20, reprezintă stilul de muzică cântat de un artist (music_genre in ('pop-rock','rock','blues','alternative','funk')).

j)MANAGER

FIRST_NAME = variabilă de tip caracter, de lungime maximă 20, reprezintă prenumele unui manager (first_name is not null).

LAST_NAME = variabilă de tip caracter, de lungime maximă 20, reprezintă numele unui manager (last_name is not null).

YEARS_EXPERIENCE = variabilă de tip întreg, de lungime maximă 10, reprezintă anii de experiență acumulați de un manager în domeniul muzical (years_experience>5).

EMAIL = variabilă de tip caracter, de lungime maximă 20, reprezintă email-ul unui manager(unique(email)).

k) EMPLOYEE_CONCERT

EMPLOYEE_ID = variabilă de tip întreg, de lungime maximă 10, cheie primară, reprezintă id-ul unui angajat.

CONCERT_ID = variabilă de tip întreg, de lungime maximă 10, cheie primară, reprezintă id-ul unui concert.

ROLE_ID = variabilă de tip întreg, de lungime maximă 10, cheie străină, reprezintă id-ul unui rol pe care îl poate avea un angajat.

GROUP_ID = variabilă de tip întreg, de lungime maximă 10, cheie străină, reprezintă id-ul unui grup din care poate face parte un angajat.

l)EMPLOYEE

FIRST_NAME = variabilă de tip caracter, de lungime maximă 20, reprezintă prenumele unui angajat (first_name is not null).

LAST_NAME = variabilă de tip caracter, de lungime maximă 20, reprezintă numele unui angajat (last_name is not null).

SALARY = variabilă de tip întreg, de lungime maximă 10, reprezintă salariul unui angajat (salary>=20).

EMAIL = variabilă de tip caracter, de lungime maximă 20, reprezintă email-ul unui angajat(unique(email)).

m) ROLE

NAME = variabilă de tip caracter, de lungime maximă 20, reprezintă numele rolului pe care îl poate avea un angajat (name is not null).

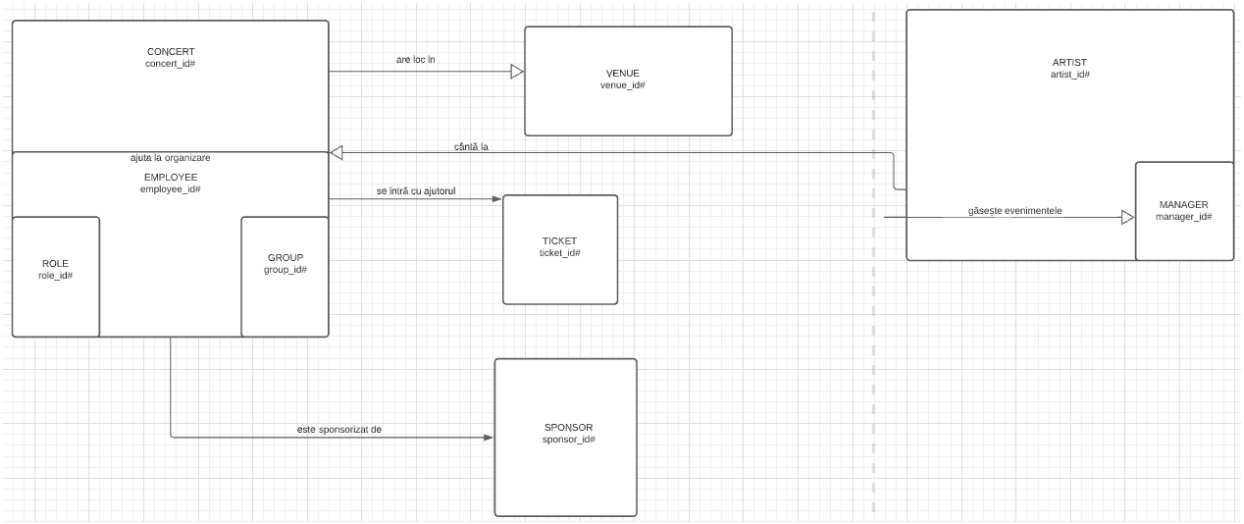
DESCRIPTION = variabilă de tip caracter, de lungime maximă 50, reprezintă descrierea rolului pe care îl poate avea un angajat.

n) GRUP

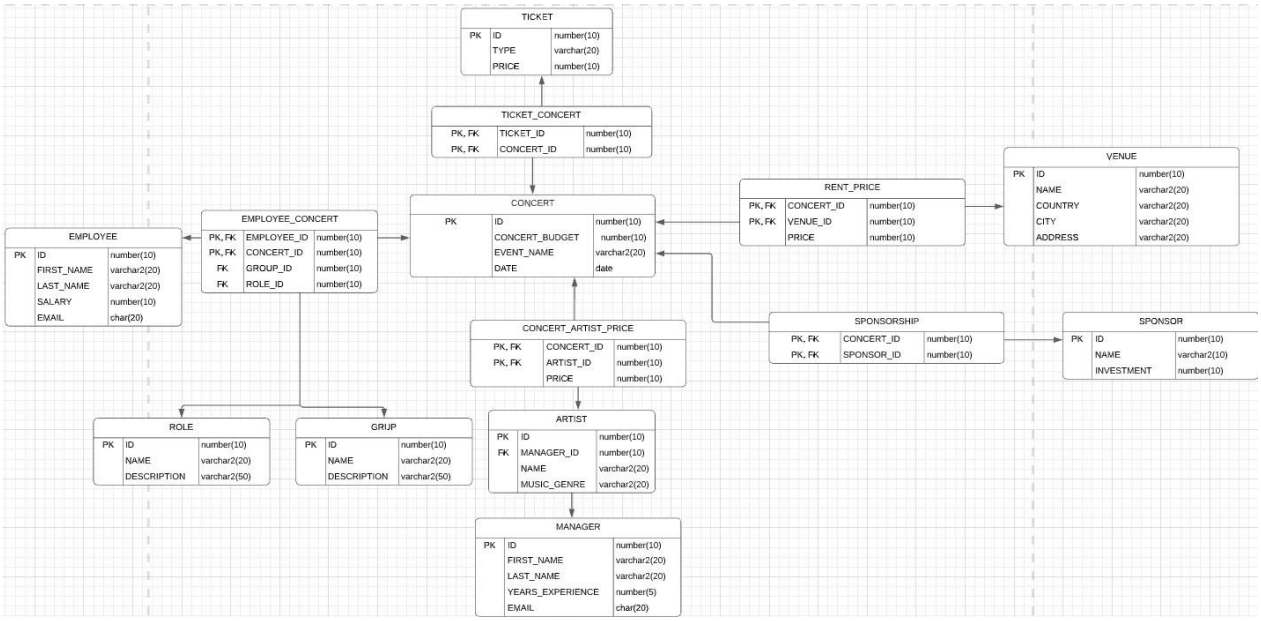
NAME = variabilă de tip caracter, de lungime maximă 20, reprezintă numele grupului din care poate face parte un angajat (name is not null).

DESCRIPTION = variabilă de tip caracter, de lungime maximă 50, reprezintă descrierea grupului din care poate face parte un angajat.

Mai jos găsiți diagrama E/R.



Mai jos găsiți diagrama conceptuală.




Schemele relaţionale corespunzătoare diagramei conceptuale din figură sunt următoarele:

-CONCERT (ID#, CONCERT_BUDGET, EVENT_NAME, DATE)

-TICKET_CONCERT (TICKET_ID#, CONCERT_ID#)

-TICKET (ID#, TYPE, PRICE)

-RENT_PRICE(CONCERT_ID#, VENUE_ID#, PRICE)

-VENUE (ID#, NAME, COUNTRY, CITY, ADDRESS)

-SPONSORSHIP (CONCERT_ID#, SPONSOR_ID#)

-SPONSOR (ID#, NAME, INVESTMENT)

-CONCERT_ARTIST_PRICE (CONCERT_ID#, ARTIST_ID#, PRICE)

-ARTIST (ID#, MANAGER_ID, NAME, MUSIC_GENRE)

-MANAGER (ID#, FIRST_NAME, LAST_NAME, YEARS_EXPERIENCE, EMAIL)

-EMPLOYEE_CONCERT (EMPLOYEE_ID#, CONCERT_ID#, ROLE_ID, GROUP_ID)

-EMPLOYEE (ID#, FIRST_NAME, LAST_NAME, SALARY, EMAIL)

-ROLE (ID#, NAME, DESCRIPTION)

-GRUP (ID#, NAME, DESCRIPTION)

Mai jos găsiti printscreen-urile cu tabelele create.

-CONCERT

| | ID | CONCERT_BUDGET | EVENT_NAME | DATE |
|---|---|---|---|---|
| 1 | 1 | 11000 | Ragan | 11-FEB-23 |
| 2 | 2 | 10500 | Ant | 03-APR-21 |
| 3 | 3 | 13500 | Deck | 17-JAN-30 |
| 4 | 4 | 11000 | Wacken | 23-AUG-22 |
| 5 | 5 | 20000 | Vast | 30-NOV-25 |
| 6 | 6 | 100000 | Temper | 29-DEC-27 |
| 7 | 7 | 14320 | Castlevania | 05-AUG-22 |
| 8 | 8 | 500000 | Ant | 01-JAN-24 |
| 9 | 9 | 20000 | Ring | 13-MAY-28 |
| 10 | 10 | 17000 | Vast | 20-DEC-21 |

-TICKET_CONCERT

| | TICKET_ID | CONCERT_ID |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 3 | 1 |
| 4 | 4 | 2 |
| 5 | 5 | 2 |
| 6 | 6 | 2 |
| 7 | 7 | 3 |
| 8 | 8 | 3 |
| 9 | 9 | 3 |
| 10 | 10 | 4 |
| 11 | 11 | 4 |
| 12 | 12 | 4 |
| 13 | 1 | 5 |
| 14 | 2 | 5 |
| 15 | 3 | 5 |
| 16 | 4 | 6 |
| 17 | 5 | 6 |
| 18 | 6 | 6 |
| 19 | 7 | 7 |
| 20 | 8 | 7 |
| 21 | 9 | 7 |
| 22 | 10 | 8 |
| 23 | 11 | 8 |
| 24 | 12 | 8 |
| 25 | 1 | 9 |
| 26 | 2 | 9 |
| 27 | 3 | 9 |
| 28 | 4 | 10 |
| 29 | 5 | 10 |
| 30 | 6 | 10 |

-TICKET

| | ID | TYPE | PRICE |
|---|---|---|---|
| 1 | 1 | adult | 50 |
| 2 | 2 | child | 20 |
| 3 | 3 | student | 30 |
| 4 | 4 | adult | 100 |
| 5 | 5 | child | 50 |
| 6 | 6 | student | 70 |
| 7 | 7 | adult | 150 |
| 8 | 8 | child | 100 |
| 9 | 9 | student | 110 |
| 10 | 10 | adult | 200 |
| 11 | 11 | child | 150 |
| 12 | 12 | student | 155 |

-RENT_PRICE

| | CONCERT_ID | VENUE_ID | PRICE |
|---|---|---|---|
| 1 | 1 | 9 | 1000 |
| 2 | 2 | 1 | 2500 |
| 3 | 3 | 2 | 1000 |
| 4 | 4 | 5 | 700 |
| 5 | 5 | 7 | 925 |
| 6 | 6 | 4 | 1200 |
| 7 | 7 | 6 | 3000 |
| 8 | 8 | 6 | 1700 |
| 9 | 9 | 10 | 1350 |
| 10 | 10 | 3 | 880 |

-VENUE

| | ID | NAME | COUNTRY | CITY | ADDRESS |
|---|---|---|---|---|---|
| 1 | 1 | Alsace | Croatia | Zagreb | 11 c |
| 2 | 2 | Lorenne | Greece | Samotraki | 7 a |
| 3 | 3 | Cast | Serbia | Belgrade | 4 k |
| 4 | 4 | Viren | Germany | Berlin | 23 d |
| 5 | 5 | Kolos | Israel | Tel Aviv | 15 b |
| 6 | 6 | Turuk | Turkey | Istanbul | 19 f |
| 7 | 7 | Nepav | Greece | Athens | 20 c |
| 8 | 8 | Antigua | Germany | Munchen | 1 a |
| 9 | 9 | Bora | Russia | Moskow | 2 b |
| 10 | 10 | Vanjt | Ukraine | Odessa | 33 g |

-SPONSORSHIP

| | CONCERT_ID | SPONSOR_ID |
|---|---|---|
| 1 | 1 | 10 |
| 2 | 1 | 9 |
| 3 | 2 | 1 |
| 4 | 2 | 7 |
| 5 | 3 | 4 |
| 6 | 3 | 5 |
| 7 | 3 | 3 |
| 8 | 4 | 2 |
| 9 | 4 | 10 |
| 10 | 5 | 7 |
| 11 | 6 | 6 |
| 12 | 6 | 4 |
| 13 | 7 | 6 |
| 14 | 8 | 2 |
| 15 | 8 | 9 |
| 16 | 8 | 3 |
| 17 | 9 | 1 |
| 18 | 9 | 4 |
| 19 | 9 | 5 |
| 20 | 10 | 8 |

-SPONSOR

| | ID | NAME | INVESTMENT |
|---|---|---|---|
| 1 | 1 | Corvus | 100 |
| 2 | 2 | Tronte | 900 |
| 3 | 3 | Markus | 1500 |
| 4 | 4 | Octavian | 125 |
| 5 | 5 | Mannstein | 100 |
| 6 | 6 | Otto | 750 |
| 7 | 7 | Erwin | 300 |
| 8 | 8 | Heinz | 1100 |
| 9 | 9 | Calin | 615 |
| 10 | 10 | Valy | 1500 |

-CONCERT_ARTIST_PRICE

| | CONCERT_ID | ARTIST_ID | PRICE |
|---|---|---|---|
| 1 | 1 | 1 | 4000 |
| 2 | 2 | 7 | 5000 |
| 3 | 3 | 5 | 3250 |
| 4 | 4 | 2 | 2700 |
| 5 | 5 | 3 | 7000 |
| 6 | 6 | 2 | 1300 |
| 7 | 7 | 10 | 6600 |
| 8 | 8 | 8 | 5000 |
| 9 | 9 | 4 | 2300 |
| 10 | 10 | 9 | 4950 |

-ARTIST

| | ID | MANAGER_ID | NAME | MUSIC_GENRE |
|---|---|---|---|---|
| 1 | 1 | 3 | Ary | blues |
| 2 | 2 | 1 | Alex | rock |
| 3 | 3 | 7 | Brugner | pop-rock |
| 4 | 4 | 2 | Ervan | funk |
| 5 | 5 | 9 | Ingus | alternative |
| 6 | 6 | 8 | Toles | blues |
| 7 | 7 | 10 | Robert | rock |
| 8 | 8 | 1 | Vunk | alternative |
| 9 | 9 | 4 | Stamp | funk |
| 10 | 10 | 5 | Wong | pop-rock |

-MANAGER

| | ID | FIRST_NAME | LAST_NAME | YEARS_EXPERIENCE | EMAIL |
|---|---|---|---|---|---|
| 1 | 1 | Alex | Lars | 12 | alex@gmail.com |
| 2 | 2 | Bruks | Norton | 6 | nort@gmail.com |
| 3 | 3 | Peter | Muller | 8 | mull@gmail.com |
| 4 | 4 | Heidrich | Hoffmann | 6 | homan@gmail.com |
| 5 | 5 | Reinhardt | Baumer | 15 | baum@gmail.com |
| 6 | 6 | Jose | Maria | 7 | maria23@gmail.com |
| 7 | 7 | Alejandro | Miguel | 9 | migus@gmail.com |
| 8 | 8 | Marco | Polo | 10 | paus7@gmail.com |
| 9 | 9 | Cristoph | Waltz | 8 | swing@gmail.com |
| 10 | 10 | Renata | Carlovsky | 11 | sky33@gmail.com |

-EMPLOYEE_CONCERT

| | EMPLOYEE_ID | CONCERT_ID | GRUP_ID | ROLE_ID |
|---|---|---|---|---|
| 1 | 10 | 1 | 5 | 4 |
| 2 | 6 | 1 | 4 | 4 |
| 3 | 4 | 1 | 5 | 1 |
| 4 | 1 | 2 | 1 | 5 |
| 5 | 7 | 2 | 3 | 2 |
| 6 | 8 | 2 | 2 | 3 |
| 7 | 6 | 2 | 3 | 3 |
| 8 | 5 | 3 | 4 | 3 |
| 9 | 2 | 3 | 4 | 2 |
| 10 | 10 | 4 | 1 | 5 |
| 11 | 9 | 4 | 2 | 4 |
| 12 | 8 | 5 | 5 | 1 |
| 13 | 1 | 5 | 3 | 3 |
| 14 | 4 | 5 | 2 | 1 |
| 15 | 7 | 6 | 1 | 2 |
| 16 | 2 | 6 | 1 | 4 |
| 17 | 9 | 7 | 5 | 2 |
| 18 | 3 | 7 | 1 | 1 |
| 19 | 5 | 7 | 4 | 3 |
| 20 | 8 | 7 | 3 | 3 |
| 21 | 9 | 8 | 4 | 5 |
| 22 | 10 | 8 | 2 | 1 |
| 23 | 4 | 8 | 1 | 4 |
| 24 | 1 | 9 | 4 | 2 |
| 25 | 5 | 9 | 1 | 3 |
| 26 | 3 | 9 | 5 | 4 |
| 27 | 3 | 10 | 5 | 4 |
| 28 | 5 | 10 | 3 | 5 |
| 29 | 7 | 10 | 2 | 1 |
| 30 | 6 | 10 | 4 | 3 |

-EMPLOYEE

| | ID | FIRST_NAME | LAST_NAME | SALARY | EMAIL |
|---|---|---|---|---|---|
| 1 | 1 | Andrei | Bordeianu | 130 | ab@gmail.com |
| 2 | 2 | Cosmin | Nedelcu | 300 | cn@gmail.com |
| 3 | 3 | Hongos | Negyem | 235 | hn@gmail.com |
| 4 | 4 | Mirela | Voicu | 115 | mv@gmail.com |
| 5 | 5 | Alex | Pintiliasa | 270 | ap@gmail.com |
| 6 | 6 | Tudor | Stavar | 130 | ts@gmail.com |
| 7 | 7 | Robert | Moisei | 300 | rm@gmail.com |
| 8 | 8 | Dragos | Popovici | 220 | dp@gmail.com |
| 9 | 9 | Florin | Patrascu | 170 | fp@gmail.com |
| 10 | 10 | Valentin | Stirbu | 420 | vs@gmail.com |

-ROLE

| | ID | NAME | DESCRIPTION |
|---|---|---|---|
| 1 | 1 | Marketing | Sells merchandise |
| 2 | 2 | Medical Assistant | Helps any injured participant |
| 3 | 3 | Web Designer | Makes an attractive website |
| 4 | 4 | Photographer | Photoshoot the funny moments |
| 5 | 5 | Construction Worker | Is setting up the concert place |

-GRUP

| | ID | NAME | DESCRIPTION |
|---|---|---|---|
| 1 | 1 | Alpha | Young members |
| 2 | 2 | Beta | Dedicated members |
| 3 | 3 | Gamma | Attractive members |
| 4 | 4 | Sigma | Hardworking members |
| 5 | 5 | Omega | Valuable members |

6) Formulați în limbaj natural o problemă pe care să orezolvați folosind unsubprogram stocatcare să utilizeze douătipuride colecție studiate. Apelați subprogramul.

```
--6
--Folositi o functie stocata care sa primeasca ca parametru un numar
--si stocheze intr-un tablou imbricat primele n valori din tabela concert
--si dupa sa stocheze intr-un vector numarul de angajati care lucreaza la primele n
--joburi si dupa sa afiseze numarul concertelor cu id-ul stocat in vector la care lucreaza un numar impar de
angajati angajati;

CREATE OR REPLACE FUNCTION f_cci
   (v_numar concert.id%TYPE)
 RETURN NUMBER IS
   numar     NUMBER(10);
   TYPE tablou_indexat IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
   t  tablou_indexat;
   TYPE vector IS VARRAY(20) OF NUMBER;
   v    vector:= vector();
BEGIN
   numar := 0;

   FOR i IN 1..v_numar LOOP
      t(i):=i;
   END LOOP;

   FOR i IN t.FIRST..t.LAST LOOP
      v.extend;
      select count(*) into v(i)
      from employee e, employee_concert ec, concert c
      where c.id = ec.concert_id and e.id = ec.employee_id
      and c.id = t(i);
   END LOOP;

   FOR i IN v.FIRST..v.LAST LOOP
      if mod(v(i),2) = 1 then
         numar := numar + 1;
      end if;
   end loop;

   return numar;

end f_cci;

BEGIN
   DBMS_OUTPUT.PUT_LINE('Numarul este '|| f_cci(10));
END;
```

```
--6
--Folositi o functie stocata care sa primeasca ca parametru un numar
--si stocheze intr-un tablou imbricat primele n valori din tabela concert
--si dupa sa stocheze intr-un vector numarul de angajati care lucreaza la primele n
--joburi si dupa sa afiseze numarul concertelor cu id-ul stocat in vector la care lucreaza un numar impar de angajati angajati;

CREATE OR REPLACE FUNCTION f_cci
    (v_numar concert.id%TYPE)
 RETURN NUMBER IS
    numar     NUMBER(10);
    TYPE tablou_indexat IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
    t  tablou_indexat;
    TYPE vector IS VARRAY(20) OF NUMBER;
    v    vector:= vector();
BEGIN
    numar := 0;

    FOR i IN 1..v_numar LOOP
        t(i):=i;
    END LOOP;

    FOR i IN t.FIRST..t.LAST LOOP
        v.extend;
        select count(*) into v(i)
        from employee e, employee_concert ec, concert c
        where c.id = ec.concert_id and e.id = ec.employee_id
        and c.id = t(i);
    END LOOP;

    FOR i IN v.FIRST..v.LAST LOOP
        if mod(v(i),2) = 1 then
            numar := numar + 1;
```

Script Output | Query Result | Query Result 1

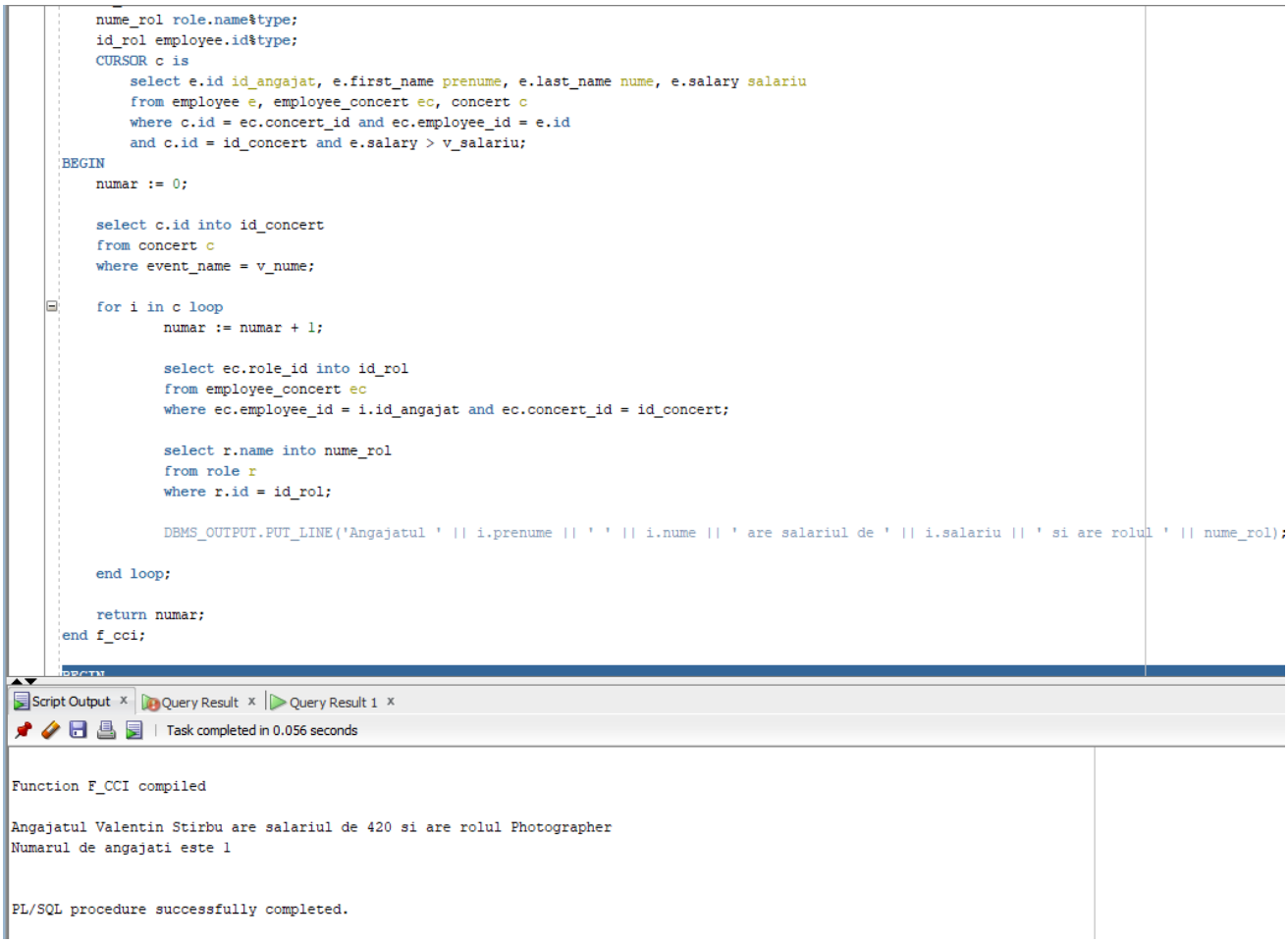
Task completed in 0.081 seconds

```
Function F_CCI compiled

Numarul este 5


PL/SQL procedure successfully completed.
```

7) Formulați în limbaj natural o problemă pe care să orezolvați folosind un subprogram stocatcare să utilizeze un tip de cursor studiat. Apelați subprogramul.

```
--7
--Folositi o functie stocata in care primeste ca parametrii
--numele unui concert si salariu si stocheaza intr-un cursor
--angajatii care lucreaza la acel concert si care au salariul
-- >= cu cel dat ca parametru si afiseaza rolul fiecarui angajat si numarul de angajati care corespund cerintelor

CREATE OR REPLACE FUNCTION f_cci
   (v_nume concert.event_name%TYPE, v_salariu employee.salary%type)
 RETURN NUMBER IS
   numar    NUMBER(10);
   id_concert concert.id%type;
   nume_rol role.name%type;
   id_rol employee.id%type;
   CURSOR c is
      select e.id id_angajat, e.first_name prenume, e.last_name nume, e.salary salariu
      from employee e, employee_concert ec, concert c
      where c.id = ec.concert_id and ec.employee_id = e.id
      and c.id = id_concert and e.salary > v_salariu;
```

```
BEGIN
   numar := 0;

   select c.id into id_concert
   from concert c
   where event_name = v_nume;

   for i in c loop
         numar := numar + 1;

         select ec.role_id into id_rol
         from employee_concert ec
         where ec.employee_id = i.id_angajat and ec.concert_id = id_concert;

         select r.name into nume_rol
         from role r
         where r.id = id_rol;

         DBMS_OUTPUT.PUT_LINE('Angajatul ' || i.prenume || ' ' || i.nume || ' are salariul de ' || i.salariu || '
si are rolul ' || nume_rol);

   end loop;

   return numar;
end f_cci;

BEGIN
   DBMS_OUTPUT.PUT_LINE('Numarul de angajati este '|| f_cci('Ragan',150));
END;
```



```
    nume_rol role.name%type;
    id_rol employee.id%type;
    CURSOR c is
        select e.id id_angajat, e.first_name prenume, e.last_name nume, e.salary salariu
        from employee e, employee_concert ec, concert c
        where c.id = ec.concert_id and ec.employee_id = e.id
        and c.id = id_concert and e.salary > v_salariu;
BEGIN
    numar := 0;

    select c.id into id_concert
    from concert c
    where event_name = v_nume;

    for i in c loop
            numar := numar + 1;

            select ec.role_id into id_rol
            from employee_concert ec
            where ec.employee_id = i.id_angajat and ec.concert_id = id_concert;

            select r.name into nume_rol
            from role r
            where r.id = id_rol;

            DBMS_OUTPUT.PUT_LINE('Angajatul ' || i.prenume || ' ' || i.nume || ' are salariul de ' || i.salariu || ' si are rolul ' || nume_rol);

    end loop;

    return numar;
end f_cci;
```

Script Output | Query Result | Query Result 1

Task completed in 0.056 seconds

```
Function F_CCI compiled

Angajatul Valentin Stirbu are salariul de 420 si are rolul Photographer
Numarul de angajati este 1


PL/SQL procedure successfully completed.
```

8) Formulați în limbaj natural o problemă pe care să orezolvați folosind un subprogram stocatde tip funcțiecare să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramulastfel încât să evidențiați toate cazurile tratate.

```
--8
--Afisati numele, bugetul, data si numarul concertelor
--care au avut loc in locatia cu numele dat ca parametru
--in functie de data

CREATE OR REPLACE FUNCTION f_cci
  (v_nume venue.name%TYPE)
 RETURN NUMBER IS
  numar    NUMBER(10);
  id_locatie venue.id%type;
  cursor c is
     select c.event_name nume, c.concert_budget buget, c.event_date data_concert
     from concert c, rent_price rp, venue v
     where rp.venue_id = id_locatie and rp.concert_id = c.id
     group by c.event_name, c.concert_budget, c.event_date;
BEGIN
  numar := 0;

  select id into id_locatie
```

```
   from venue
   where name = v_nume;

   DBMS_OUTPUT.PUT_LINE('La locatia cu numele ' || v_nume);

   for i in c loop
      numar := numar + 1;

      if i.data_concert > sysdate then
         DBMS_OUTPUT.PUT_LINE('Va avea loc ' || i.nume || ' cu bugetul ' || i.buget || ' si la data ' ||
i.data_concert);
      else
         DBMS_OUTPUT.PUT_LINE('A avut loc ' || i.nume || ' cu bugetul ' || i.buget || ' si la data ' ||
i.data_concert);
      end if;

   end loop;

   return numar;

   EXCEPTION

   WHEN NO_DATA_FOUND THEN
       RAISE_APPLICATION_ERROR(-20000,'Nu exista concerte care au loc in locatia data ca parametru');
   WHEN TOO_MANY_ROWS THEN
      RAISE_APPLICATION_ERROR(-20001,'Exista prea multe concerte la locatia data ca parametru');
   WHEN OTHERS THEN
      RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');

end f_cci;


BEGIN
   DBMS_OUTPUT.PUT_LINE('Numarul de concerte este '|| f_cci('Turuk'));
END;
```

```
    select id into id_locatie
    from venue
    where name = v_nume;

    DBMS_OUTPUT.PUT_LINE('La locatia cu numele ' || v_nume);

    for i in c loop
        numar := numar + 1;

        if i.data_concert > sysdate then
            DBMS_OUTPUT.PUT_LINE('Va avea loc ' || i.nume || ' cu bugetul ' || i.buget || ' si la data ' || i.data_concert);
        else
            DBMS_OUTPUT.PUT_LINE('A avut loc ' || i.nume || ' cu bugetul ' || i.buget || ' si la data ' || i.data_concert);
        end if;

    end loop;

    return numar;

    EXCEPTION

    WHEN NO_DATA_FOUND THEN
         RAISE_APPLICATION_ERROR(-20000,'Nu exista concerte care au loc in locatia data ca parametru');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001,'Exista prea multe concerte la locatia data ca parametru');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');

end f_cci;


BEGIN
    DBMS_OUTPUT.PUT_LINE('Numarul de concerte este ' || f_cci('Turuk'));
END;
```

Script Output | Query Result | Query Result 1

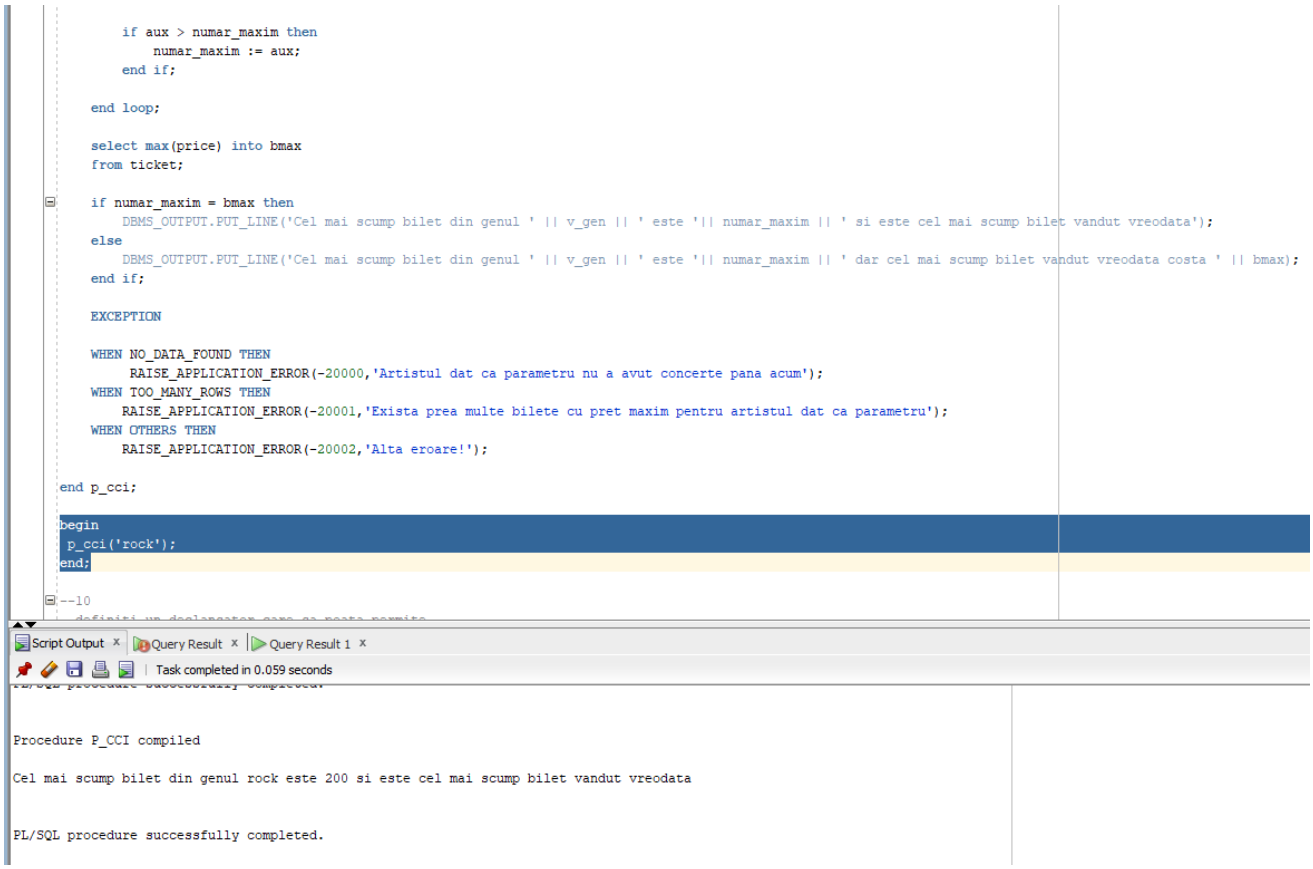
Task completed in 0.078 seconds

```
Function F_CCI compiled

La locatia cu numele Turuk
Va avea loc Ant cu bugetul 500000 si la data 01-JAN-24
Va avea loc Castlevania cu bugetul 14320 si la data 05-AUG-22
Numarul de concerte este 2


PL/SQL procedure successfully completed.
```

9) Formulați în limbaj natural o problemă pe care să orezolvați folosind un subprogram stocat de tip procedurăcare să utilizezeîntr-o singură comandă SQL5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramulastfel încât să evidențiați toate cazurile tratate.

```
--9
--Afisati pretul celui mai scump bilet
--pe care l-a vandut un artist care
--canta un gen de muzica dat ca parametru
--si verifica daca este cel mai scump bilet
--vandut vreodata sau nu

CREATE OR REPLACE PROCEDURE p_cci
   (v_gen artist.music_genre%TYPE)
IS
   aux NUMBER(10);
   numar_maxim    NUMBER(10);
   bmax number(10);
   cursor c is
      select a.id id_art
      from artist a
```

```
      where music_genre = v_gen;
BEGIN
   numar_maxim := 0;
   aux := 0;

   for i in c loop
      select max(t.price) into aux
      from ticket t, ticket_concert tc, concert c, concert_artist_price cap
      where i.id_art = cap.artist_id and cap.concert_id = c.id and tc.concert_id = c.id and tc.ticket_id = t.id;

      if aux > numar_maxim then
         numar_maxim := aux;
      end if;

   end loop;

   select max(price) into bmax
   from ticket;

   if numar_maxim = bmax then
      DBMS_OUTPUT.PUT_LINE('Cel mai scump bilet din genul ' || v_gen || ' este '|| numar_maxim || ' si
este cel mai scump bilet vandut vreodata');
   else
      DBMS_OUTPUT.PUT_LINE('Cel mai scump bilet din genul ' || v_gen || ' este '|| numar_maxim || ' dar
cel mai scump bilet vandut vreodata costa ' || bmax);
   end if;

   EXCEPTION

   WHEN NO_DATA_FOUND THEN
       RAISE_APPLICATION_ERROR(-20000,'Artistul dat ca parametru nu a avut concerte pana acum');
   WHEN TOO_MANY_ROWS THEN
      RAISE_APPLICATION_ERROR(-20001,'Exista prea multe bilete cu pret maxim pentru artistul dat ca
parametru');
   WHEN OTHERS THEN
      RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');

end p_cci;

begin
 p_cci('rock');
end;
```




10) Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.

--10

--definiti un declansator care sa poata permite

--lucrul asupra bazei de data doar daca o face

--in zi impara de la ora 10 la 12 si numele utilizatorului este Mark

```
CREATE OR REPLACE TRIGGER t_cci
   BEFORE INSERT OR UPDATE OR DELETE ON sponsor
BEGIN
   IF MOD(TO_NUMBER(TO_CHAR(sysdate,'DD')),2) != 1 then
      RAISE_APPLICATION_ERROR(-20000,'Tabelul nu poate fi actualizat deoarece nu este zi impara');
   elsif (TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 10 AND 12) then
      RAISE_APPLICATION_ERROR(-20001,'Tabelul nu poate fi actualizat deoarece nu se afla in intervalul orar
10 - 12');
   elsif USER != UPPER('mark') THEN
```

RAISE_APPLICATION_ERROR(-20002,'Tabelul nu poate fi actualizat deoarece utilizatorul nu este Mark');

END IF;

END;

update sponsor set investment = investment + 50;

```
--10
--definiti un declansator care sa poata permite
--lucrul asupra bazei de data doar daca o face
--in zi impara de la ora 10 la 12 si numele utilizatorului este Mark

CREATE OR REPLACE TRIGGER t_cci
    BEFORE INSERT OR UPDATE OR DELETE ON sponsor
BEGIN
    IF MOD(TO_NUMBER(TO_CHAR(sysdate,'DD')),2) != 1 then
        RAISE_APPLICATION_ERROR(-20000,'Tabelul nu poate fi actualizat deoarece nu este zi impara');
    elsif (TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 10 AND 12) then
        RAISE_APPLICATION_ERROR(-20001,'Tabelul nu poate fi actualizat deoarece nu se afla in intervalul orar 10 - 12');
    elsif USER != UPPER('mark') THEN
        RAISE_APPLICATION_ERROR(-20002,'Tabelul nu poate fi actualizat deoarece utilizatorul nu este Mark');
    END IF;
END;

update sponsor set investment = investment + 50;

--11
--definiti un declansator la nivel de linie
--care sa nu permita stergerea salariilor angajatilor
--care se afla in limita intervalului marginit de
--cel mai mic salariu si cel mai mare salariu
--sa nu permita modificarea salariilor maxime si minime
--si sa nu permita inserarea in tabel

create or replace trigger tl_cci
    before update or delete or insert of salary on employee
    for each row
declare
```

```
Script Output x | Query Result x | Query Result 1 x
Task completed in 0.072 seconds

Error starting at line : 217 in command -
update sponsor set investment = investment + 50
Error report -
ORA-20001: Tabelul nu poate fi actualizat deoarece nu se afla in intervalul orar 10 - 12
ORA-06512: at "CALIN.T_CCI", line 5
ORA-04088: error during execution of trigger 'CALIN.T_CCI'
```

11) Definiți un triggerde tip LMD la nivel de linie.Declanșați trigger-ul.

--definiti un declansator la nivel de linie
--care sa nu permita stergerea salariilor angajatilor
--care se afla in limita intervalului marginit de
--cel mai mic salariu si cel mai mare salariu
--sa nu permita modificarea salariilor maxime si minime
--si sa nu permita inserarea in tabel

```
create or replace trigger t1_cci
   before update or delete or insert of salary on employee
   for each row
declare
   sal_min employee.salary%type;
   sal_max employee.salary%type;
begin

   select max(salary), min(salary)
   into sal_max, sal_min
   from employee;

   if deleting and :old.salary > sal_min and :old.salary < sal_max then
      RAISE_APPLICATION_ERROR(-20000,'Salariu nu poate fi sters deoarece nu este maxim sau minim');
   elsif updating and :old.salary = sal_min then
      RAISE_APPLICATION_ERROR(-20001,'Salariu nu poate fi actualizat deoarece este salariul minim');
   elsif updating and :old.salary = sal_max then
      RAISE_APPLICATION_ERROR(-20002,'Salariu nu poate fi actualizat deoarece este salariul maxim');
   elsif inserting then
      RAISE_APPLICATION_ERROR(-20003,'Nu se poate insera in tabel');
   end if;
end;


update employee set salary = salary + 50 where salary = 420;
```

```
    for each row
declare
    sal_min employee.salary%type;
    sal_max employee.salary%type;
begin

    select max(salary), min(salary)
    into sal_max, sal_min
    from employee;

    if deleting and :old.salary > sal_min and :old.salary < sal_max then
        RAISE_APPLICATION_ERROR(-20000,'Salariu nu poate fi sters deoarece nu este maxim sau minim');
    elsif updating and :old.salary = sal_min then
        RAISE_APPLICATION_ERROR(-20001,'Salariu nu poate fi actualizat deoarece este salariul minim');
    elsif updating and :old.salary = sal_max then
        RAISE_APPLICATION_ERROR(-20002,'Salariu nu poate fi actualizat deoarece este salariul maxim');
    elsif inserting then
        RAISE_APPLICATION_ERROR(-20003,'Nu se poate insera in tabel');
    end if;
end;


update employee set salary = salary + 50 where salary = 420;

--12
--creati un tabel ce va detine detalii despre user
--si populati liniile cu detalii cu ajutorul unu trigger
--dupa ce utilizatorul foloseste o comanda ldd de adaugare a
--unei noi coloane la tabela employee

CREATE TABLE user_cci (
    user_name          VARCHAR2(30),
```

Script Output | Query Result | Query Result 1

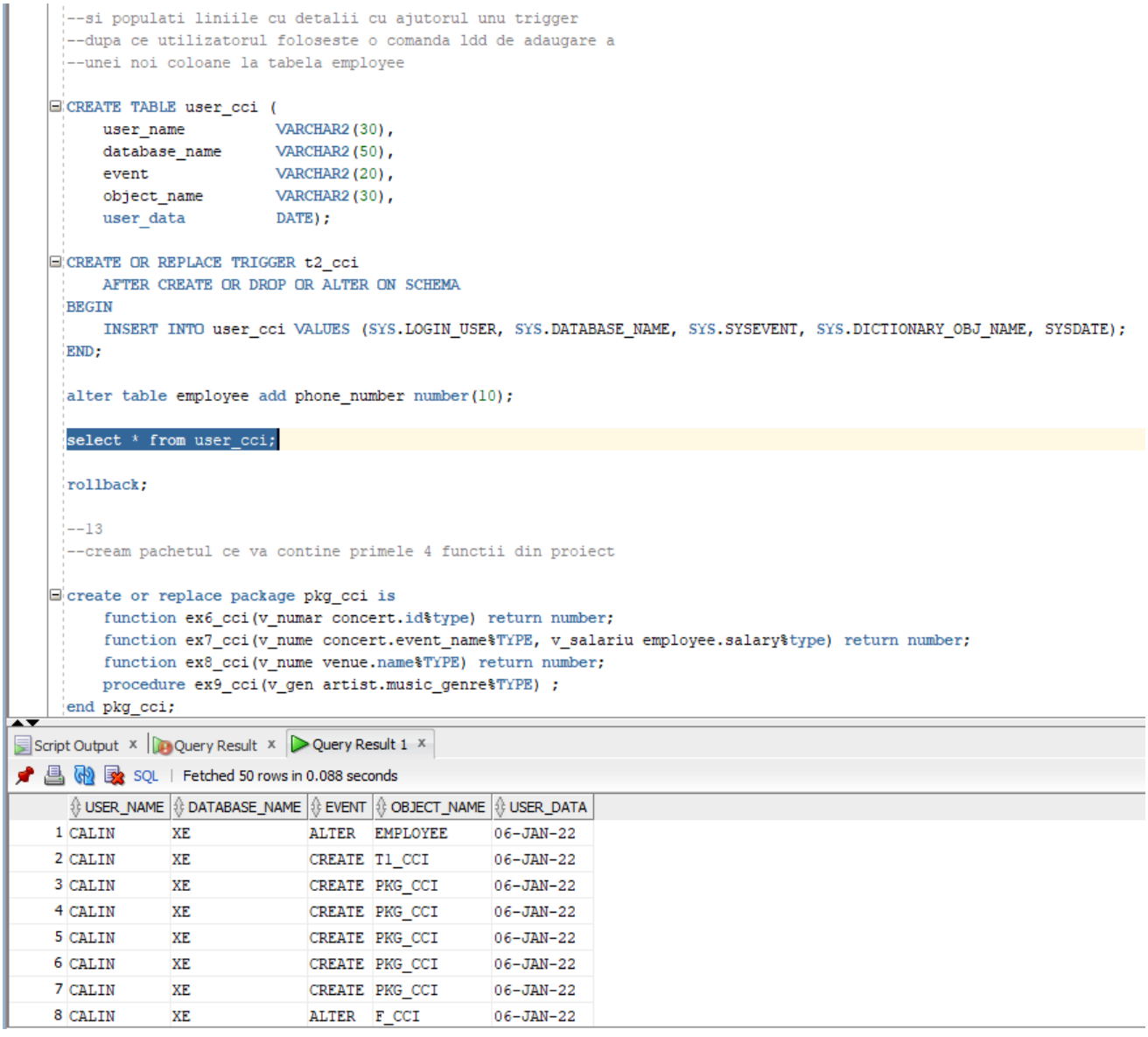
Task completed in 0.096 seconds

```
Error starting at line : 251 in command -
update employee set salary = salary + 50 where salary = 420
Error report -
ORA-04091: table CALIN.EMPLOYEE is mutating, trigger/function may not see it
ORA-06512: at "CALIN.T1_CCI", line 6
ORA-04088: error during execution of trigger 'CALIN.T1_CCI'
```

12) Definiți un triggerde tip LDD. Declanșați trigger-ul.

```
--12
--creati un tabel ce va detine detalii despre user
--si populati liniile cu detalii cu ajutorul unu trigger
--dupa ce utilizatorul foloseste o comanda ldd de adaugare a
--unei noi coloane la tabela employee

CREATE TABLE user_cci (
   user_name        VARCHAR2(30),
   database_name    VARCHAR2(50),
   event            VARCHAR2(20),
   object_name      VARCHAR2(30),
   user_data        DATE);
```

```
CREATE OR REPLACE TRIGGER t2_cci
   AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
   INSERT INTO user_cci VALUES (SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT,
SYS.DICTIONARY_OBJ_NAME, SYSDATE);
END;

alter table employee add phone_number number(10);

select * from user_cci;

rollback;
```



```
--si populati liniile cu detalii cu ajutorul unu trigger
--dupa ce utilizatorul foloseste o comanda ldd de adaugare a
--unei noi coloane la tabela employee

CREATE TABLE user_cci (
    user_name          VARCHAR2(30),
    database_name      VARCHAR2(50),
    event              VARCHAR2(20),
    object_name        VARCHAR2(30),
    user_data          DATE);

CREATE OR REPLACE TRIGGER t2_cci
    AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
    INSERT INTO user_cci VALUES (SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSDATE);
END;

alter table employee add phone_number number(10);

select * from user_cci;

rollback;

--13
--cream pachetul ce va contine primele 4 functii din proiect

create or replace package pkg_cci is
    function ex6_cci(v_numar concert.id%type) return number;
    function ex7_cci(v_nume concert.event_name%TYPE, v_salariu employee.salary%type) return number;
    function ex8_cci(v_nume venue.name%TYPE) return number;
    procedure ex9_cci(v_gen artist.music_genre%TYPE) ;
end pkg_cci;
```

Script Output | Query Result | Query Result 1

SQL | Fetched 50 rows in 0.088 seconds

| | USER_NAME | DATABASE_NAME | EVENT | OBJECT_NAME | USER_DATA |
|---|---|---|---|---|---|
| 1 | CALIN | XE | ALTER | EMPLOYEE | 06-JAN-22 |
| 2 | CALIN | XE | CREATE | T1_CCI | 06-JAN-22 |
| 3 | CALIN | XE | CREATE | PKG_CCI | 06-JAN-22 |
| 4 | CALIN | XE | CREATE | PKG_CCI | 06-JAN-22 |
| 5 | CALIN | XE | CREATE | PKG_CCI | 06-JAN-22 |
| 6 | CALIN | XE | CREATE | PKG_CCI | 06-JAN-22 |
| 7 | CALIN | XE | CREATE | PKG_CCI | 06-JAN-22 |
| 8 | CALIN | XE | ALTER | F_CCI | 06-JAN-22 |

13) Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.

--13
--cream pachetul ce va contine primele 4 functii din proiect

```
create or replace package pkg_cci is
   function ex6_cci(v_numar concert.id%type) return number;
   function ex7_cci(v_nume concert.event_name%TYPE, v_salariu employee.salary%type) return number;
   function ex8_cci(v_nume venue.name%TYPE) return number;
   procedure ex9_cci(v_gen artist.music_genre%TYPE) ;
end pkg_cci;

create or replace package body pkg_cci is

function ex6_cci(v_numar concert.id%type)
   return number is
   numar     NUMBER(10);
   TYPE tablou_indexat IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
   t  tablou_indexat;
   TYPE vector IS VARRAY(20) OF NUMBER;
   v    vector:= vector();
BEGIN
   numar := 0;

   FOR i IN 1..v_numar LOOP
      t(i):=i;
   END LOOP;

   FOR i IN t.FIRST..t.LAST LOOP
      v.extend;
      select count(*) into v(i)
      from employee e, employee_concert ec, concert c
      where c.id = ec.concert_id and e.id = ec.employee_id
      and c.id = t(i);
   END LOOP;

   FOR i IN v.FIRST..v.LAST LOOP
      if mod(v(i),2) = 1 then
         numar := numar + 1;
      end if;
   end loop;

   return numar;

end ex6_cci;

function ex7_cci(v_nume concert.event_name%TYPE, v_salariu employee.salary%type)
   return number is
   numar     NUMBER(10);
   id_concert concert.id%type;
   nume_rol role.name%type;
   id_rol employee.id%type;
   CURSOR c is
      select e.id id_angajat, e.first_name prenume, e.last_name nume, e.salary salariu
      from employee e, employee_concert ec, concert c
      where c.id = ec.concert_id and ec.employee_id = e.id
      and c.id = id_concert and e.salary > v_salariu;
```

```
BEGIN
   numar := 0;

   select c.id into id_concert
   from concert c
   where event_name = v_nume;

   for i in c loop
         numar := numar + 1;

         select ec.role_id into id_rol
         from employee_concert ec
         where ec.employee_id = i.id_angajat and ec.concert_id = id_concert;

         select r.name into nume_rol
         from role r
         where r.id = id_rol;

         DBMS_OUTPUT.PUT_LINE('Angajatul ' || i.prenume || ' ' || i.nume || ' are salariul de ' || i.salariu || '
si are rolul ' || nume_rol);

   end loop;

   return numar;
end ex7_cci;

function ex8_cci(v_nume venue.name%TYPE)
   return number is
   numar     NUMBER(10);
   id_locatie venue.id%type;
   cursor c is
      select c.event_name nume, c.concert_budget buget, c.event_date data_concert
      from concert c, rent_price rp, venue v
      where rp.venue_id = id_locatie and rp.concert_id = c.id
      group by c.event_name, c.concert_budget, c.event_date;
BEGIN
   numar := 0;

   select id into id_locatie
   from venue
   where name = v_nume;

   DBMS_OUTPUT.PUT_LINE('La locatia cu numele ' || v_nume);

   for i in c loop
      numar := numar + 1;

      if i.data_concert > sysdate then
         DBMS_OUTPUT.PUT_LINE('Va avea loc ' || i.nume || ' cu bugetul ' || i.buget || ' si la data ' ||
i.data_concert);
      else
         DBMS_OUTPUT.PUT_LINE('A avut loc' || i.nume || ' cu bugetul ' || i.buget || ' si la data ' ||
```

```
i.data_concert);
      end if;

   end loop;

   return numar;

   EXCEPTION

   WHEN NO_DATA_FOUND THEN
       RAISE_APPLICATION_ERROR(-20000,'Nu exista concerte care au loc in locatia data ca parametru');
   WHEN TOO_MANY_ROWS THEN
      RAISE_APPLICATION_ERROR(-20001,'Exista prea multe concerte la locatia data ca  parametru');
   WHEN OTHERS THEN
      RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');

end ex8_cci;

procedure ex9_cci(v_gen artist.music_genre%TYPE) IS
   aux NUMBER(10);
   numar_maxim     NUMBER(10);
   bmax number(10);
   cursor c is
      select a.id id_art
      from artist a
      where music_genre = v_gen;
BEGIN
   numar_maxim := 0;
   aux := 0;

   for i in c loop
      select max(t.price) into aux
      from ticket t, ticket_concert tc, concert c, concert_artist_price cap
      where i.id_art = cap.artist_id and cap.concert_id = c.id and tc.concert_id = c.id and tc.ticket_id = t.id;

      if aux > numar_maxim then
         numar_maxim := aux;
      end if;

   end loop;

   select max(price) into bmax
   from ticket;

   if numar_maxim = bmax then
      DBMS_OUTPUT.PUT_LINE('Cel mai scump bilet din genul ' || v_gen || ' este '|| numar_maxim || ' si
este cel mai scump bilet vandut vreodata');
   else
      DBMS_OUTPUT.PUT_LINE('Cel mai scump bilet din genul ' || v_gen || ' este '|| numar_maxim || ' dar
cel mai scump bilet vandut vreodata costa ' || bmax);
   end if;
```

```
   EXCEPTION

   WHEN NO_DATA_FOUND THEN
       RAISE_APPLICATION_ERROR(-20000,'Artistul dat ca parametru nu a avut concerte pana acum');
   WHEN TOO_MANY_ROWS THEN
      RAISE_APPLICATION_ERROR(-20001,'Exista prea multe bilete cu pret maxim pentru artistul dat ca
parametru');
   WHEN OTHERS THEN
      RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');

end ex9_cci;

end pkg_cci;

begin
   DBMS_OUTPUT.PUT_LINE('Numarul este ' || pkg_cci.ex6_cci(10));
end;

BEGIN
   DBMS_OUTPUT.PUT_LINE('Numarul de angajati este '|| pkg_cci.ex7_cci('Ragan',150));
END;

BEGIN
   DBMS_OUTPUT.PUT_LINE('Numarul de concerte este '|| pkg_cci.ex8_cci('Turuk'));
END;

begin
 pkg_cci.ex9_cci('rock');
end;
```

```
Package PKG_CCI compiled


Package Body PKG_CCI compiled

Numarul este 5


PL/SQL procedure successfully completed.

Angajatul Valentin Stirbu are salariul de 420 si are rolul Photographer
Numarul de angajati este 1


PL/SQL procedure successfully completed.

La locatia cu numele Turuk
Va avea loc Ant cu bugetul 500000 si la data 01-JAN-24
Va avea loc Castlevania cu bugetul 14320 si la data 05-AUG-22
Numarul de concerte este 2


PL/SQL procedure successfully completed.

Cel mai scump bilet din genul rock este 200 si este cel mai scump bilet vandut vreodata


PL/SQL procedure successfully completed.
```

14) Definiți un pachet care să includătipuri de date complexe și obiecte necesare unui flux deacțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2funcții, minim 2 proceduri).

```
--14
--prima functie => pune intr-un tablou id-urile pare din intervalul [x,y] date ca parametru si returneaza
numarul de angajati care au salariul > decat salariul mediu
--prima procedura => afiseaza numele grupului in care lucreaza cei mai multi angajati si daca angajatul cu
numele dat ca parametru face parte din grup
--a doua functie => returneaza 1 daca angajatul cu numele dat lucreaza in grupul respectiv si 0 in caz contrar
--a foua procedura => uneste functiile si afiseaza

create or replace package Final_cci is
   function salariatiPesteMedie(v_x employee.id%TYPE, v_y employee.id%TYPE) return number;
   procedure maxGrup(nume employee.last_name%type, prenume employee.first_name%type);
   function verifAng(id_grup grup.id%type, nume employee.last_name%type, prenume
employee.first_name%type) return number;
   procedure afisare(lim_inf number, lim_sup number, nume employee.last_name%type, prenume
employee.first_name%type);
end Final_cci;


create or replace package body Final_cci is

FUNCTION salariatiPesteMedie
   (v_x employee.id%TYPE, v_y employee.id%TYPE)
```

```
 RETURN NUMBER IS
   numar    NUMBER(10);
   TYPE vector IS VARRAY(20) OF NUMBER;
   v   vector:= vector();
   TYPE tablou_indexat IS TABLE OF employee%rowtype INDEX BY PLS_INTEGER;
   t  tablou_indexat;
   j number;
   media number;
BEGIN
   j := 0;
   numar := 0;

   select avg(salary) into media
   from employee;

   for i in v_x..v_y loop
      if mod(i,2) = 0 then
         j := j + 1;
         v.extend;
         v(j) := i;
      end if;
   end loop;

   for i in v.first..v.last loop
      select *
      bulk collect into t
      from employee e
      where e.salary > media and e.id = v(i);
   end loop;

   for i in t.first..t.last loop
      numar := numar + 1;
   end loop;

   return numar;
END salariatiPesteMedie;

procedure maxGrup(nume employee.last_name%type, prenume employee.first_name%type) is
   nr_max number;
   nume_grup varchar2(20);
   aux number;
   bool_ang number;
begin
   bool_ang := 0;

   select max(count(g.id)) into nr_max
   from grup g, employee_concert ec, employee e
   where g.id = ec.grup_id and ec.employee_id = e.id
   group by g.id;

   for i in 1..5 loop
```

```
      select max(count(g.id)) into aux
      from grup g, employee_concert ec, employee e
      where ec.grup_id = i and ec.employee_id = e.id
      group by g.id;

      if aux = nr_max then

         select g.name into nume_grup
         from grup g
         where g.id =  i;

         bool_ang := verifAng(i, nume, prenume);

      end if;

   end loop;


   if bool_ang = 1 then
      DBMS_OUTPUT.PUT_LINE('Grupul cu numar maxim de angajati este '|| nume_grup || ' si angajatul ' ||
nume || ' ' || prenume || ' face parte din grup');
   end if;

end maxGrup;

function verifAng(id_grup grup.id%type, nume employee.last_name%type , prenume
employee.first_name%type)
   return number is
   numar number;
   id_ang number;
   nume_grup varchar2(20);
   cursor c is
      select ec.employee_id idAng
      from employee_concert ec
      where grup_id = id_grup;
begin
   numar := 0;

   select g.name into nume_grup
   from grup g
   where g.id = id_grup;

   select e.id into id_ang
   from employee e
   where e.first_name = prenume and e.last_name = nume;

   for i in c loop
      if i.idAng = id_ang then
         numar := 1;
      else continue;
      end if;
   end loop;
```

```
    return numar;


EXCEPTION

    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Grupul cu numar maxim de angajati este '|| nume_grup || ' si angajatul ' ||
nume || ' ' || prenume || ' nu face parte din grup');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001,'Exista prea multe angajati cu acest nume in grupul respectiv');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');

end verifAng;

procedure afisare(lim_inf number, lim_sup number, nume employee.last_name%type, prenume
employee.first_name%type) is
    numar number;
begin

    numar := Final_cci.salariatiPesteMedie(lim_inf, lim_sup);

    if numar = 1 then
        DBMS_OUTPUT.PUT_LINE('In intervalul [' || lim_inf || ',' || lim_sup ||']' || ' lucreaza ' || numar || '
angajat care are id-ul par si are salariul peste medie');
    elsif numar = 0 or numar > 1 then
        DBMS_OUTPUT.PUT_LINE('In intervalul [' || lim_inf || ',' || lim_sup ||']' || ' lucreaza ' || numar || '
angajati care au id-ul par si au salariul peste medie');
    end if;

    Final_cci.maxGrup(nume,prenume);

end afisare;

end Final_cci;


begin
    Final_cci.afisare(1,10,'Bordeianu','Andrei');
end;
```

```
Package FINAL_CCI compiled


Package Body FINAL_CCI compiled

In intervalul [1,10] lucreaza 1 angajat care are id-ul par si are salariul peste medie
Grupul cu numar maxim de angajati este Alpha si angajatul Bordeianu Andrei face parte din grup


PL/SQL procedure successfully completed.
```