Sistem de gestiune a activității hoteliere

Lăutaru Bianca-Maria

Seria 25, Grupa 252

Anul Universitar 2025-2026

Cuprins

Introducere

Acest proiect constă în proiectarea și implementarea unei baze de date relaționale pentru gestiunea unui hotel. Scopul principal al aplicației este gestionarea eficientă a informațiilor necesare funcționării unui lanț hotelier, asigurând integritatea și consistența datelor.

Implementarea a fost realizată folosind **Oracle Database 21c Express Edition (XE)**, iar interacțiunea cu baza de date s-a realizat prin intermediul aplicației **Oracle SQL Developer**, pe un sistem de operare **Windows 11**, fără utilizarea unei mașini virtuale, baza de date fiind instalată local.

Cerința 1

Prezentați pe scurt baza de date (utilitatea ei).

1. Descrierea modelului

Modelul descrie structura unui sistem utilizat pentru gestionarea activităților unui hotel, care permite tratarea unor situații complexe întâlnite în funcționarea acestuia.

Acesta include entitățile esențiale pentru organizarea unui hotel: **Hoteluri, Angajați, Joburi, Camere, Rezervări, Facturi, Servicii, Clienți**. Fiecare entitate conține atributele necesare pentru desfășurarea eficientă a activității hotelului.

Relațiile dintre entități reflectă modul real de funcționare a unui hotel: un hotel are angajați care ocupă diferite joburi, clienții pot rezerva camere și pot adăuga servicii suplimentare, iar pentru fiecare rezervare se generează facturile asociate.

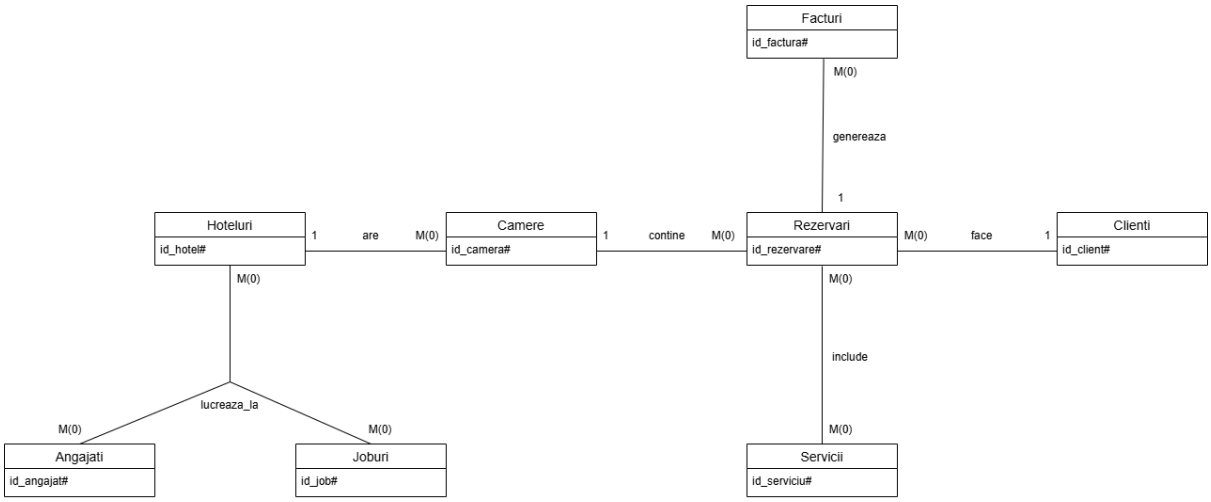
2. Regulile modelului (cardinalitățile)

- Relația **lucrează_la** (Angajați – Joburi – Hoteluri): M(0) – M(0) – M(0)
    - Un angajat poate ocupa mai multe joburi la mai multe hoteluri de-a lungul timpului sau poate să nu aibă încă niciun job sau hotel asociate.
    - Un job poate fi ocupat de mai mulți angajați din mai multe hoteluri sau de niciunul, în niciun hotel.
    - Un hotel poate avea mai mulți angajați și mai multe tipuri de joburi sau niciun angajat și niciun job asociate încă.
    - Relația devine tabelul asociativ *Istoric_Joburi* în diagrama conceptuală.
- Relația **are** (Hoteluri – Camere): 1 – M(0)
    - Un hotel poate avea mai multe camere sau niciuna (de exemplu, dacă este în renovare).
    - O cameră aparține unui singur hotel, dar nu poate exista fără să fie asociată unui hotel.
- Relația **conține** (Rezervări – Camere): M(0) – 1
    - O rezervare poate conține maxim o cameră (dacă un client dorește să rezerve mai multe camere, se vor realiza mai multe rezervări), dar nu poate să nu conțină nicio cameră.

    - O cameră poate să facă parte din mai multe rezervări (în perioade diferite) sau niciuna.
- Relația **generează** (Rezervări – Facturi): 1 – M(0)
    - O rezervare poate genera mai multe facturi (pentru avans, restul sumei de achitat, servicii suplimentare), dar poate să nu fie generată nicio factură imediat ce se face o rezervare.
    - O factură este generată de o singură rezervare, dar nu poate să existe fără aceasta.
- Relația **face** (Clienți – Rezervări): 1 – M(0)
    - Un client poate să facă mai multe rezervări sau niciuna (clientul poate să fie înregistrat în sistem, dar nu a făcut încă nicio rezervare).
    - O rezervare aparține unui singur client și nu poate exista fără client.
- Relația **include** (Rezervări – Servicii): M(0) – M(0)
    - O rezervare poate include mai multe servicii suplimentare sau niciunul.
    - Un serviciu poate să fie inclus în mai multe rezervări sau niciuna.
    - Relația devine tabelul asociativ *Servicii_Rezervări* în diagrama conceptuală.

Cerința 2

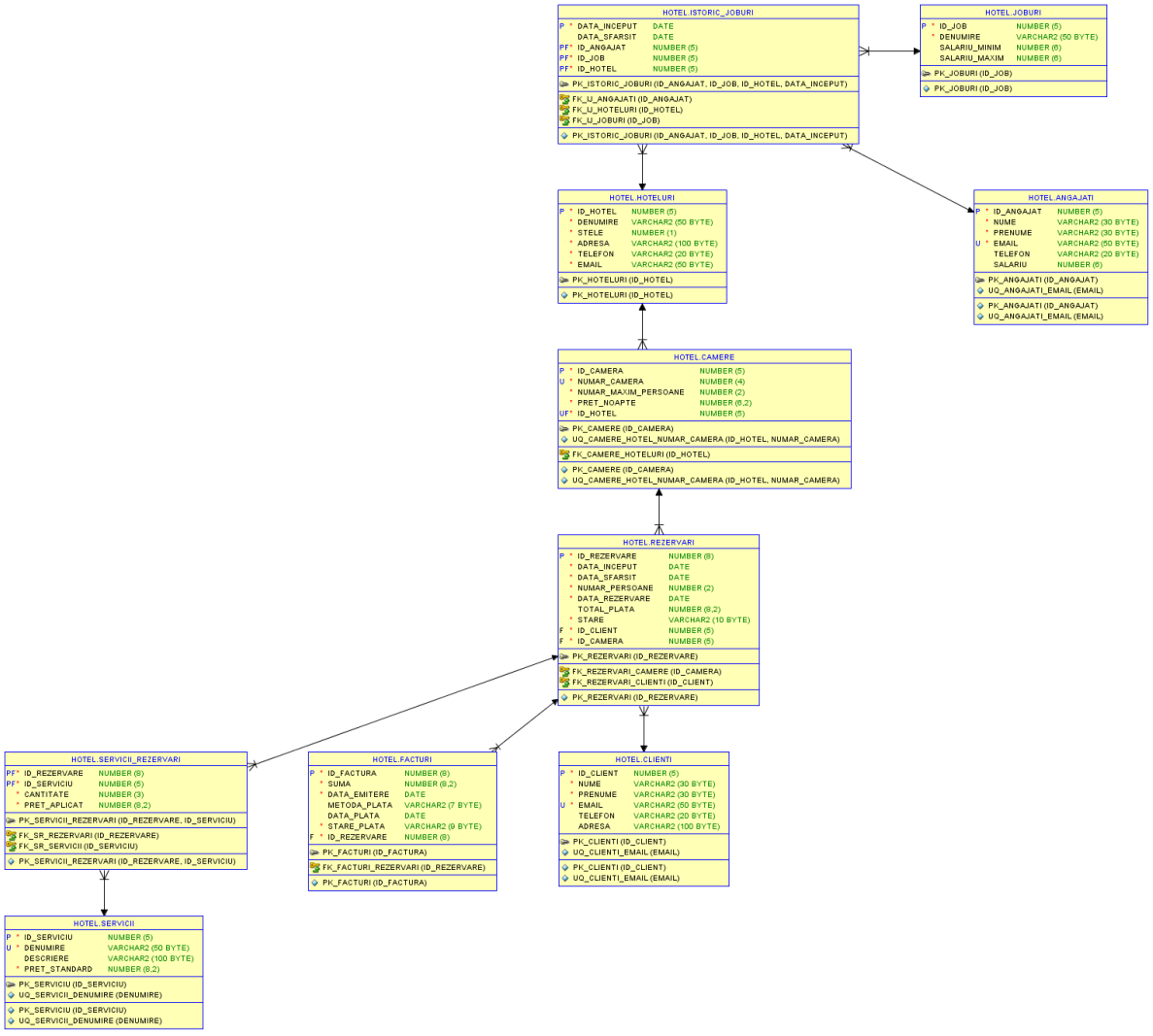
Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba română (vezi curs SGBD, model de diagramă entitate-relație; nu se va accepta alt format).



Cerința 3

Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare: entitățile, relațiile și atributele trebuie definite în limba română.

Diagrama generată în SQL Developer:



Cerința 4

Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, adăugând toate constrângerile de integritate necesare (chei primare, cheile externe etc).

```
DROP TABLE hoteluri CASCADE CONSTRAINTS;

DROP TABLE angajati CASCADE CONSTRAINTS;

DROP TABLE joburi CASCADE CONSTRAINTS;

DROP TABLE istoric_joburi CASCADE CONSTRAINTS;

DROP TABLE camere CASCADE CONSTRAINTS;

DROP TABLE clienti CASCADE CONSTRAINTS;

DROP TABLE rezervari CASCADE CONSTRAINTS;

DROP TABLE facturi CASCADE CONSTRAINTS;

DROP TABLE servicii CASCADE CONSTRAINTS;

DROP TABLE servicii_rezervari CASCADE CONSTRAINTS;


CREATE TABLE hoteluri (
    id_hotel NUMBER(5),
    denumire VARCHAR2(50) NOT NULL,
    stele NUMBER(1) NOT NULL,
    adresa VARCHAR2(100) NOT NULL,
    telefon VARCHAR2(20) NOT NULL,
    email VARCHAR2(50) NOT NULL,
    CONSTRAINT pk_hoteluri PRIMARY KEY (id_hotel),
    CONSTRAINT chk_hoteluri_stele CHECK (stele BETWEEN 1 AND 5)
);
```

```
CREATE TABLE angajati (
    id_angajat NUMBER(5),
    nume VARCHAR2(30) NOT NULL,
    prenume VARCHAR2(30) NOT NULL,
    email VARCHAR2(50) NOT NULL,
    telefon VARCHAR2(20),
    salariu NUMBER(6),
    CONSTRAINT pk_angajati PRIMARY KEY (id_angajat),
    CONSTRAINT uq_angajati_email UNIQUE (email)
);

CREATE TABLE joburi (
    id_job NUMBER(5),
    denumire VARCHAR2(50) NOT NULL,
    salariu_minim NUMBER(6),
    salariu_maxim NUMBER(6),
    CONSTRAINT pk_joburi PRIMARY KEY (id_job),
    CONSTRAINT chk_joburi_salarii CHECK (salariu_maxim IS NULL
OR salariu_minim IS NULL OR salariu_maxim >= salariu_minim)
);

CREATE TABLE istoric_joburi (
    data_inceput DATE,
    data_sfarsit DATE,
```

```
    id_angajat NUMBER(5),
    id_job NUMBER(5),
    id_hotel NUMBER(5),
    CONSTRAINT pk_istoric_joburi PRIMARY KEY (id_angajat, id_job, id_hotel, data_inceput),
    CONSTRAINT fk_ij_angajati FOREIGN KEY (id_angajat) REFERENCES angajati(id_angajat),
    CONSTRAINT fk_ij_joburi FOREIGN KEY (id_job) REFERENCES joburi(id_job),
    CONSTRAINT fk_ij_hoteluri FOREIGN KEY (id_hotel) REFERENCES hoteluri(id_hotel),
    CONSTRAINT chk_ij_perioada CHECK (data_sfarsit IS NULL OR data_inceput <= data_sfarsit)
);

CREATE TABLE camere (
    id_camera NUMBER(5),
    numar_camera NUMBER(4) NOT NULL,
    numar_maxim_persoane NUMBER(2) NOT NULL,
    pret_noapte NUMBER(6, 2) NOT NULL,
    id_hotel NUMBER(5) NOT NULL,
    CONSTRAINT pk_camere PRIMARY KEY (id_camera),
    CONSTRAINT fk_camere_hoteluri FOREIGN KEY (id_hotel) REFERENCES hoteluri(id_hotel),
    CONSTRAINT uq_camere_hotel_numar_camera UNIQUE (id_hotel, numar_camera),
```

```
    CONSTRAINT chk_camere_persoane CHECK (numar_maxim_persoane
> 0),
    CONSTRAINT chk_camere_pret CHECK (pret_noapte > 0)
);

CREATE TABLE clienti (
    id_client NUMBER(5),
    nume VARCHAR2(30) NOT NULL,
    prenume VARCHAR2(30) NOT NULL,
    email VARCHAR2(50) NOT NULL,
    telefon VARCHAR2(20),
    adresa VARCHAR2(100),
    CONSTRAINT pk_clienti PRIMARY KEY (id_client),
    CONSTRAINT uq_clienti_email UNIQUE (email)
);

CREATE TABLE rezervari (
    id_rezervare NUMBER(8),
    data_inceput DATE NOT NULL,
    data_sfarsit DATE NOT NULL,
    numar_persoane NUMBER(2) NOT NULL,
    data_rezervare DATE NOT NULL,
    total_plata NUMBER(8, 2),
    stare VARCHAR2(10) DEFAULT 'CREATA' NOT NULL,
    id_client NUMBER(5) NOT NULL,
```

```
    id_camera NUMBER(5) NOT NULL,
    CONSTRAINT pk_rezervari PRIMARY KEY (id_rezervare),
    CONSTRAINT fk_rezervari_clienti FOREIGN KEY (id_client)
REFERENCES clienti(id_client),
    CONSTRAINT fk_rezervari_camere FOREIGN KEY (id_camera)
REFERENCES camere(id_camera),
    CONSTRAINT chk_rezervare_perioada CHECK (data_inceput <
data_sfarsit),
    CONSTRAINT chk_rezervare_persoane CHECK (numar_persoane >
0),
    CONSTRAINT chk_rezervare_data CHECK (data_rezervare <=
data_inceput),
    CONSTRAINT chk_rezervari_stare CHECK (UPPER(stare) IN
('CREATA', 'CONFIRMATA', 'ANULATA', 'FINALIZATA'))
);

CREATE TABLE facturi (
    id_factura NUMBER(8),
    suma NUMBER(8, 2) NOT NULL,
    data_emitere DATE NOT NULL,
    metoda_plata VARCHAR2(7),
    data_plata DATE,
    stare_plata VARCHAR2(9) DEFAULT 'NEPLATITA' NOT NULL,
    id_rezervare NUMBER(8) NOT NULL,
    CONSTRAINT pk_facturi PRIMARY KEY (id_factura),
    CONSTRAINT fk_facturi_rezervari FOREIGN KEY (id_rezervare)
REFERENCES rezervari(id_rezervare),
```

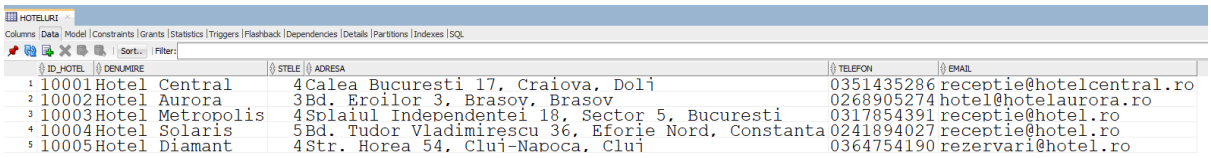
```
    CONSTRAINT chk_factura_data CHECK (data_plata IS NULL OR
data_plata >= data_emitere),

    CONSTRAINT chk_factura_metoda CHECK (UPPER(metoda_plata) IN
('CARD', 'NUMERAR', 'OP')),

    CONSTRAINT chk_factura_stare CHECK (UPPER(stare_plata) IN
('NEPLATITA', 'PARTIAL', 'PLATITA', 'ANULATA')),

    CONSTRAINT chk_factura_plata CHECK(UPPER(stare_plata) <>
'PLATITA' OR (UPPER(stare_plata) = 'PLATITA' AND data_plata IS
NOT NULL))

);


CREATE TABLE servicii (

    id_serviciu NUMBER(5),

    denumire VARCHAR2(50) NOT NULL,

    descriere VARCHAR2(100),

    pret_standard NUMBER(8, 2) NOT NULL,

    CONSTRAINT pk_serviciu PRIMARY KEY (id_serviciu),

    CONSTRAINT uq_servicii_denumire UNIQUE (denumire),

    CONSTRAINT chk_servicii_pret CHECK (pret_standard >= 0)

);


CREATE TABLE servicii_rezervari (

    id_rezervare NUMBER(8),

    id_serviciu NUMBER(5),

    cantitate NUMBER(3) DEFAULT 1 NOT NULL,

    pret_aplicat NUMBER(8,2) NOT NULL,
```

```
    CONSTRAINT pk_servicii_rezervari PRIMARY KEY (id_rezervare,
id_serviciu),

    CONSTRAINT fk_sr_rezervari FOREIGN KEY (id_rezervare)
REFERENCES rezervari(id_rezervare),

    CONSTRAINT fk_sr_servicii FOREIGN KEY (id_serviciu)
REFERENCES servicii(id_serviciu),

    CONSTRAINT chk_sr_cantitate CHECK (cantitate > 0),

    CONSTRAINT chk_sr_pret CHECK (pret_aplicat >=0)

);
```

Cerința 5

Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru fiecare tabelă asociativă).

```
INSERT INTO hoteluri VALUES(10001, 'Hotel Central', 4, 'Calea
Bucuresti 17, Craiova, Dolj', '0351435286',
'receptie@hotelcentral.ro');

INSERT INTO hoteluri VALUES(10002, 'Hotel Aurora', 3, 'Bd.
Eroilor 3, Brasov, Brasov', '0268905274',
'hotel@hotelaurora.ro');

INSERT INTO hoteluri VALUES(10003, 'Hotel Metropolis', 4,
'Splaiul Independentei 18, Sector 5, Bucuresti', '0317854391',
'receptie@hotel.ro');

INSERT INTO hoteluri VALUES(10004, 'Hotel Solaris', 5, 'Bd.
Tudor Vladimirescu 36, Eforie Nord, Constanta', '0241894027',
'receptie@hotel.ro');

INSERT INTO hoteluri VALUES(10005, 'Hotel Diamant', 4, 'Str.
Horea 54, Cluj-Napoca, Cluj', '0364754190',
'rezervari@hotel.ro');
```



HOTELURI

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort... | Filter:

	ID_HOTEL	DENUMIRE	STELE	ADRESA	TELEFON	EMAIL
1	10001	Hotel Central	4	Calea Bucuresti 17, Craiova, Dolj	0351435286	receptie@hotelcentral.ro
2	10002	Hotel Aurora	3	Bd. Eroilor 3, Brasov, Brasov	0268905274	hotel@hotelaurora.ro
3	10003	Hotel Metropolis	4	Splaiul Independentei 18, Sector 5, Bucuresti	0317854391	receptie@hotel.ro
4	10004	Hotel Solaris	5	Bd. Tudor Vladimirescu 36, Eforie Nord, Constanta	0241894027	receptie@hotel.ro
5	10005	Hotel Diamant	4	Str. Horea 54, Cluj-Napoca, Cluj	0364754190	rezervari@hotel.ro

```
INSERT INTO angajati VALUES(20001, 'Popescu', 'Ion',
'ion.popescu@gmail.com', '0745298053', 9200);

INSERT INTO angajati VALUES(20002, 'Ionescu', 'Ana',
'anaionescu@yahoo.com', '0723069824', 5500);

INSERT INTO angajati VALUES(20003, 'Dumitru', 'Andrei',
'andreidumitru@gmail.com', '0723475841', 6700);
```

```
INSERT INTO angajati VALUES(20004, 'Lazar', 'Alexandru',
'alexandru.lazar@yahoo.com', '0737539219', 8800);

INSERT INTO angajati VALUES(20005, 'Enache', 'Ioana',
'ioanaenache@gmail.com', '0736106282', 3700);

INSERT INTO angajati VALUES(20006, 'Barbu', 'Mihai',
'mihaibarbu@gmail.com', '0756829741', 5200);

INSERT INTO angajati VALUES(20007, 'Grigore', 'Elena',
'elena.grigore@gmail.com', '073520132', 3400);

INSERT INTO angajati VALUES(20008, 'Vasilescu', 'Maria',
'mariavasilescu@yahoo.com', '0746894524', 9400);

INSERT INTO angajati VALUES(20009, 'Negru', 'Andreea',
'andreeanegru@gmail.com', '0774580013', 5800);

INSERT INTO angajati VALUES(20010, 'Dumitrescu', 'Bogdan',
'bogdan.dumitrescu@yahoo.com', '0786794872', 5600);

INSERT INTO angajati VALUES(20011, 'Preda', 'Alexandra',
'alexandra.preda@gmail.com', '0747310283', 4000);

INSERT INTO angajati VALUES(20012, 'Costache', 'Teodor',
'teodorcostache@gmail.com', '0775810384', 6500);

INSERT INTO angajati VALUES(20013, 'Serban', 'Ionel',
'ionelserban@gmail.com', '0795829831', 4800);

INSERT INTO angajati VALUES(20014, 'Badea', 'Sebastian',
'sebastian.badea@yahoo.com', '0759498502', 4700);

INSERT INTO angajati VALUES(20015, 'Munteanu', 'Simona',
'simonamunteanu@gmail.com', '0757893213', 3600);

INSERT INTO angajati VALUES(20016, 'Radulescu', 'Lucian',
'lucian.radulescu@gmail.com', '0750049256', 3500);
```

ANGAJATI

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter:

	ID_ANGAJAT	NUME	PRENUME	EMAIL	TELEFON	SALARIU
1	20001	Popescu	Ion	ion.popescu@gmail.com	0745298053	9200
2	20002	Ionescu	Ana	anaionescu@yahoo.com	0723069824	5500
3	20003	Dumitru	Andrei	andreidumitru@gmail.com	0723475841	6700
4	20004	Lazar	Alexandru	alexandru.lazar@yahoo.com	0737539219	8800
5	20005	Enache	Ioana	ioanaenache@gmail.com	0736106282	3700
6	20006	Barbu	Mihai	mihaibarbu@gmail.com	0756829741	5200
7	20007	Grigore	Elena	elena.grigore@gmail.com	073520132	3400
8	20008	Vasilescu	Maria	mariavasilescu@yahoo.com	0746894524	9400
9	20009	Negru	Andreea	andreeanegru@gmail.com	0774580013	5800
10	20010	Dumitrescu	Bogdan	bogdan.dumitrescu@yahoo.com	0786794872	5600
11	20011	Preda	Alexandra	alexandra.preda@gmail.com	0747310283	4000
12	20012	Costache	Teodor	teodorcostache@gmail.com	0775810384	6500
13	20013	Serban	Ionel	ionelserban@gmail.com	0795829831	4800
14	20014	Badea	Sebastian	sebastian.badea@yahoo.com	0759498502	4700
15	20015	Munteanu	Simona	simonamunteanu@gmail.com	0757893213	3600
16	20016	Radulescu	Lucian	lucian.radulescu@gmail.com	0750049256	3500

```
INSERT INTO joburi VALUES(30001, 'Manager', 7000, 10000);

INSERT INTO joburi VALUES(30002, 'Asistent manager', 6000, 8000);

INSERT INTO joburi VALUES(30003, 'Receptioner', 4000, 6000);

INSERT INTO joburi VALUES(30004, 'Camerista', 3000, 4000);

INSERT INTO joburi VALUES(30005, 'Bucatar', 5000, 7000);

INSERT INTO joburi VALUES(30006, 'Barman', 3000, 5000);

INSERT INTO joburi VALUES(30007, 'Tehnician mentenanta', 3000, 4000);

INSERT INTO joburi VALUES(30008, 'Administrator IT', 5000, 8000);
```

JOBURI

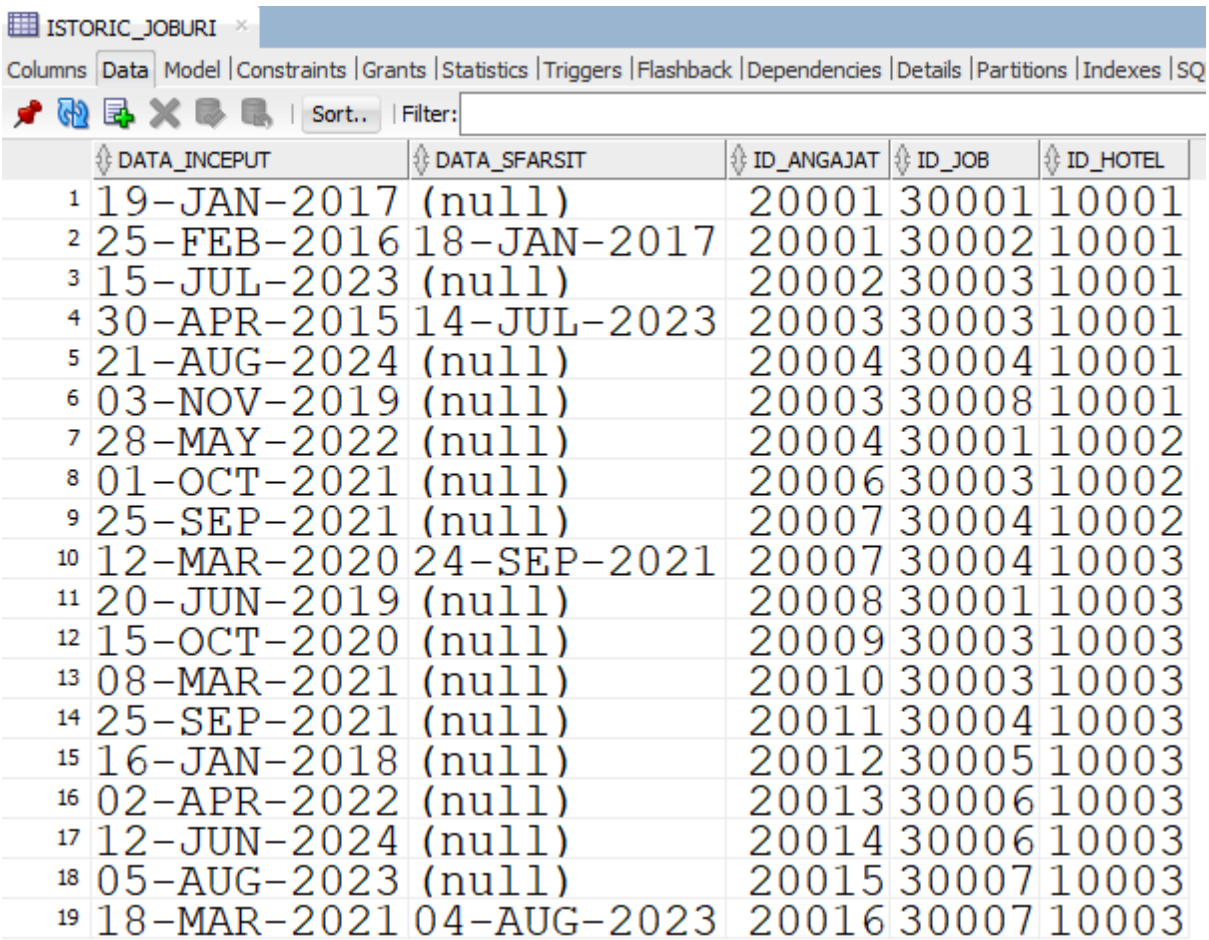
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter:

	ID_JOB	DENUMIRE	SALARIU_MINIM	SALARIU_MAXIM
1	30001	Manager	7000	10000
2	30002	Asistent manager	6000	8000
3	30003	Receptioner	4000	6000
4	30004	Camerista	3000	4000
5	30005	Bucatar	5000	7000
6	30006	Barman	3000	5000
7	30007	Tehnician mentenanta	3000	4000
8	30008	Administrator IT	5000	8000

```
INSERT INTO istoric_joburi VALUES(TO_DATE('19-01-2017', 'DD-MM-
YYYY'), NULL , 20001, 30001, 10001);

INSERT INTO istoric_joburi VALUES(TO_DATE('25-02-2016', 'DD-MM-
YYYY'),  TO_DATE('18-01-2017',  'DD-MM-YYYY'),  20001,  30002,
10001);

INSERT INTO istoric_joburi VALUES(TO_DATE('15-07-2023', 'DD-MM-
YYYY'), NULL, 20002, 30003, 10001);

INSERT INTO istoric_joburi VALUES(TO_DATE('30-04-2015', 'DD-MM-
YYYY'),  TO_DATE('14-07-2023',  'DD-MM-YYYY'),  20003,  30003,
10001);

INSERT INTO istoric_joburi VALUES(TO_DATE('21-08-2024', 'DD-MM-
YYYY'), NULL, 20004, 30004, 10001);

INSERT INTO istoric_joburi VALUES(TO_DATE('03-11-2019', 'DD-MM-
YYYY'), NULL, 20003, 30008, 10001);

INSERT INTO istoric_joburi VALUES(TO_DATE('28-05-2022', 'DD-MM-
YYYY'), NULL, 20004, 30001, 10002);

INSERT INTO istoric_joburi VALUES(TO_DATE('01-10-2021', 'DD-MM-
YYYY'), NULL, 20006, 30003, 10002);

INSERT INTO istoric_joburi VALUES(TO_DATE('25-09-2021', 'DD-MM-
YYYY'), NULL, 20007, 30004, 10002);

INSERT INTO istoric_joburi VALUES(TO_DATE('12-03-2020', 'DD-MM-
YYYY'),  TO_DATE('24-09-2021',  'DD-MM-YYYY'),  20007,  30004,
10003);

INSERT INTO istoric_joburi VALUES(TO_DATE('20-06-2019', 'DD-MM-
YYYY'), NULL, 20008, 30001, 10003);

INSERT INTO istoric_joburi VALUES(TO_DATE('15-10-2020', 'DD-MM-
YYYY'), NULL, 20009, 30003, 10003);

INSERT INTO istoric_joburi VALUES(TO_DATE('08-03-2021', 'DD-MM-
YYYY'), NULL, 20010, 30003, 10003);
```

```
INSERT INTO istoric_joburi VALUES(TO_DATE('25-09-2021', 'DD-MM-YYYY'), NULL, 20011, 30004, 10003);

INSERT INTO istoric_joburi VALUES(TO_DATE('16-01-2018', 'DD-MM-YYYY'), NULL, 20012, 30005, 10003);

INSERT INTO istoric_joburi VALUES(TO_DATE('02-04-2022', 'DD-MM-YYYY'), NULL, 20013, 30006, 10003);

INSERT INTO istoric_joburi VALUES(TO_DATE('12-06-2024', 'DD-MM-YYYY'), NULL, 20014, 30006, 10003);

INSERT INTO istoric_joburi VALUES(TO_DATE('05-08-2023', 'DD-MM-YYYY'), NULL, 20015, 30007, 10003);

INSERT INTO istoric_joburi VALUES(TO_DATE('18-03-2021', 'DD-MM-YYYY'), TO_DATE('04-08-2023', 'DD-MM-YYYY'), 20016, 30007, 10003);
```



ISTORIC_JOBURI

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQ

Sort.. | Filter:

	DATA_INCEPUT	DATA_SFARSIT	ID_ANGAJAT	ID_JOB	ID_HOTEL
1	19-JAN-2017	(null)	20001	30001	10001
2	25-FEB-2016	18-JAN-2017	20001	30002	10001
3	15-JUL-2023	(null)	20002	30003	10001
4	30-APR-2015	14-JUL-2023	20003	30003	10001
5	21-AUG-2024	(null)	20004	30004	10001
6	03-NOV-2019	(null)	20003	30008	10001
7	28-MAY-2022	(null)	20004	30001	10002
8	01-OCT-2021	(null)	20006	30003	10002
9	25-SEP-2021	(null)	20007	30004	10002
10	12-MAR-2020	24-SEP-2021	20007	30004	10003
11	20-JUN-2019	(null)	20008	30001	10003
12	15-OCT-2020	(null)	20009	30003	10003
13	08-MAR-2021	(null)	20010	30003	10003
14	25-SEP-2021	(null)	20011	30004	10003
15	16-JAN-2018	(null)	20012	30005	10003
16	02-APR-2022	(null)	20013	30006	10003
17	12-JUN-2024	(null)	20014	30006	10003
18	05-AUG-2023	(null)	20015	30007	10003
19	18-MAR-2021	04-AUG-2023	20016	30007	10003

```
INSERT INTO camere VALUES(40001, 101, 1, 300, 10001);
INSERT INTO camere VALUES(40002, 102, 2, 450, 10001);
INSERT INTO camere VALUES(40003, 103, 2, 450, 10001);
INSERT INTO camere VALUES(40004, 201, 3, 550, 10001);
INSERT INTO camere VALUES(40005, 202, 2, 500, 10001);
INSERT INTO camere VALUES(40006, 301, 4, 650, 10001);
INSERT INTO camere VALUES(40007, 302, 4, 700, 10001);
INSERT INTO camere VALUES(40008, 101, 1, 250, 10002);
INSERT INTO camere VALUES(40009, 102, 1, 250, 10002);
INSERT INTO camere VALUES(40010, 201, 2, 400, 10002);
INSERT INTO camere VALUES(40011, 202, 2, 400, 10002);
INSERT INTO camere VALUES(40012, 203, 2, 500, 10002);
INSERT INTO camere VALUES(40013, 301, 3, 550, 10002);
INSERT INTO camere VALUES(40014, 302, 4, 600, 10002);
INSERT INTO camere VALUES(40015, 101, 2, 500, 10003);
INSERT INTO camere VALUES(40016, 102, 2, 500, 10003);
INSERT INTO camere VALUES(40017, 103, 2, 550, 10003);
INSERT INTO camere VALUES(40018, 104, 2, 550, 10003);
INSERT INTO camere VALUES(40019, 201, 3, 600, 10003);
INSERT INTO camere VALUES(40020, 202, 3, 600, 10003);
INSERT INTO camere VALUES(40021, 203, 4, 750, 10003);
INSERT INTO camere VALUES(40022, 204, 4, 750, 10003);
INSERT INTO camere VALUES(40023, 205, 4, 800, 10003);
```

CAMERE

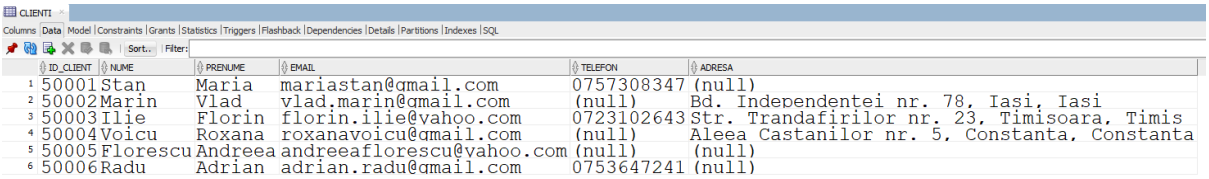
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | In

Sort.. | Filter:

	ID_CAMERA	NUMAR_CAMERA	NUMAR_MAXIM_PERSOANE	PRET_NOAPTE	ID_HOTEL
1	40001	101	1	300	10001
2	40002	102	2	450	10001
3	40003	103	2	450	10001
4	40004	201	3	550	10001
5	40005	202	2	500	10001
6	40006	301	4	650	10001
7	40007	302	4	700	10001
8	40008	101	1	250	10002
9	40009	102	1	250	10002
10	40010	201	2	400	10002
11	40011	202	2	400	10002
12	40012	203	2	500	10002
13	40013	301	3	550	10002
14	40014	302	4	600	10002
15	40015	101	2	500	10003
16	40016	102	2	500	10003
17	40017	103	2	550	10003
18	40018	104	2	550	10003
19	40019	201	3	600	10003
20	40020	202	3	600	10003
21	40021	203	4	750	10003
22	40022	204	4	750	10003
23	40023	205	4	800	10003

```
INSERT INTO clienti VALUES(50001, 'Stan', 'Maria',
'mariastan@gmail.com', '0757308347', NULL);

INSERT INTO clienti VALUES(50002, 'Marin', 'Vlad',
'vlad.marin@gmail.com', NULL, 'Bd. Independenței nr. 78, Iasi,
Iasi');

INSERT INTO clienti VALUES(50003, 'Ilie', 'Florin',
'florin.ilie@yahoo.com', '0723102643', 'Str. Trandafirilor nr.
23, Timisoara, Timis');
```

```
INSERT INTO clienti VALUES(50004, 'Voicu', 'Roxana',
'roxanavoicu@gmail.com', NULL, 'Aleea Castanilor nr. 5,
Constanta, Constanta');

INSERT INTO clienti VALUES(50005, 'Florescu', 'Andreea',
'andreeaflorescu@yahoo.com', NULL, NULL);

INSERT INTO clienti VALUES(50006, 'Radu', 'Adrian',
'adrian.radu@gmail.com', '0753647241', NULL);
```

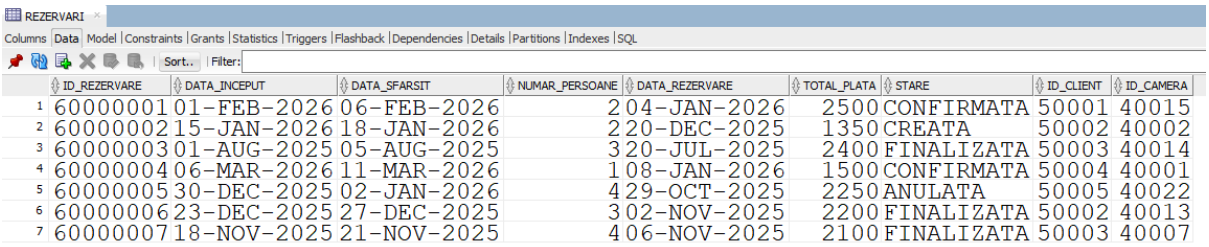


CLIENTI

	ID_CLIENT	NUME	PRENUME	EMAIL	TELEFON	ADRESA
1	50001	Stan	Maria	mariastan@gmail.com	0757308347	(null)
2	50002	Marin	Vlad	vlad.marin@gmail.com	(null)	Bd. Independentei nr. 78, Iasi, Iasi
3	50003	Ilie	Florin	florin.ilie@yahoo.com	0723102643	Str. Trandafirilor nr. 23, Timisoara, Timis
4	50004	Voicu	Roxana	roxanavoicu@gmail.com	(null)	Aleea Castanilor nr. 5, Constanta, Constanta
5	50005	Florescu	Andreea	andreeaflorescu@yahoo.com	(null)	(null)
6	50006	Radu	Adrian	adrian.radu@gmail.com	0753647241	(null)

```
INSERT INTO rezervari VALUES (60000001, TO_DATE('01-02-2026',
'DD-MM-YYYY'), TO_DATE('06-02-2026', 'DD-MM-YYYY'), 2,
TO_DATE('04-01-2026', 'DD-MM-YYYY'), 2500, 'CONFIRMATA', 50001,
40015);

INSERT INTO rezervari VALUES (60000002, TO_DATE('15-01-2026',
'DD-MM-YYYY'), TO_DATE('18-01-2026', 'DD-MM-YYYY'), 2,
TO_DATE('20-12-2025', 'DD-MM-YYYY'), 1350, 'CREATA', 50002,
40002);

INSERT INTO rezervari VALUES (60000003, TO_DATE('01-08-2025',
'DD-MM-YYYY'), TO_DATE('05-08-2025', 'DD-MM-YYYY'), 3,
TO_DATE('20-07-2025', 'DD-MM-YYYY'), 2400, 'FINALIZATA', 50003,
40014);

INSERT INTO rezervari VALUES (60000004, TO_DATE('06-03-2026',
'DD-MM-YYYY'), TO_DATE('11-03-2026', 'DD-MM-YYYY'), 1,
TO_DATE('08-01-2026', 'DD-MM-YYYY'), 1500, 'CONFIRMATA', 50004,
40001);

INSERT INTO rezervari VALUES (60000005, TO_DATE('30-12-2025',
'DD-MM-YYYY'), TO_DATE('02-01-2026', 'DD-MM-YYYY'), 4,
```

```
TO_DATE('29-10-2025', 'DD-MM-YYYY'), 2250, 'ANULATA', 50005,
40022);

INSERT INTO rezervari VALUES (60000006, TO_DATE('23-12-2025',
'DD-MM-YYYY'), TO_DATE('27-12-2025', 'DD-MM-YYYY'), 3,
TO_DATE('02-11-2025', 'DD-MM-YYYY'), 2200, 'FINALIZATA', 50002,
40013);

INSERT INTO rezervari VALUES (60000007, TO_DATE('18-11-2025',
'DD-MM-YYYY'), TO_DATE('21-11-2025', 'DD-MM-YYYY'), 4,
TO_DATE('06-11-2025', 'DD-MM-YYYY'), 2100, 'FINALIZATA', 50003,
40007);
```



REZERVARI

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

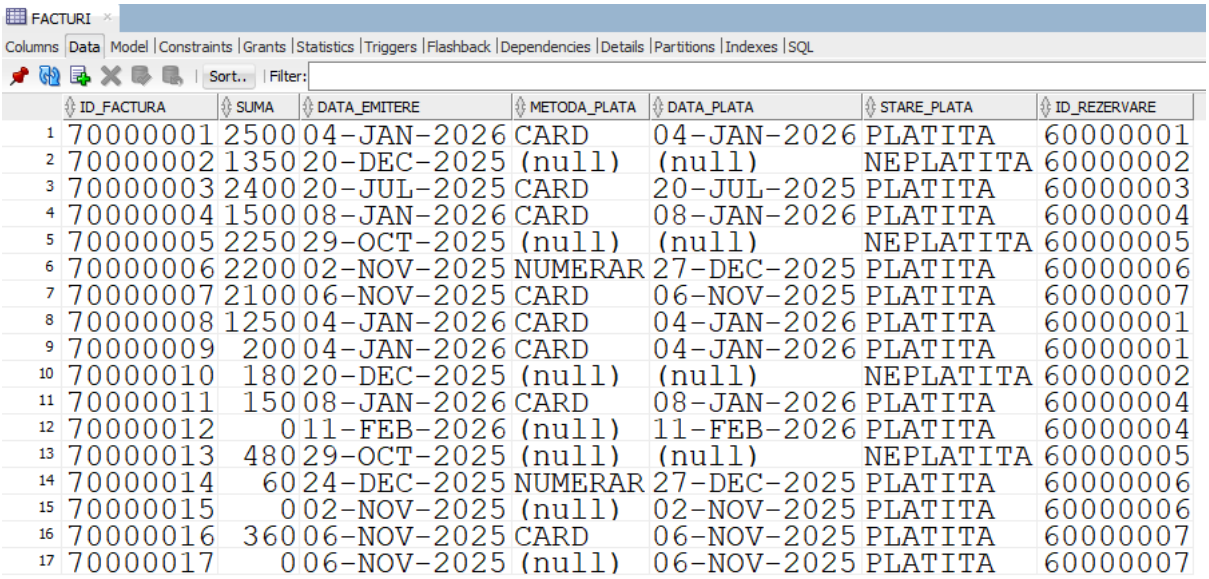
Sort.. | Filter:

	ID_REZERVARE	DATA_INCEPUT	DATA_SFARSIT	NUMAR_PERSOANE	DATA_REZERVARE	TOTAL_PLATA	STARE	ID_CLIENT	ID_CAMERA
1	60000001	01-FEB-2026	06-FEB-2026	2	04-JAN-2026	2500	CONFIRMATA	50001	40015
2	60000002	15-JAN-2026	18-JAN-2026	2	20-DEC-2025	1350	CREATA	50002	40002
3	60000003	01-AUG-2025	05-AUG-2025	3	20-JUL-2025	2400	FINALIZATA	50003	40014
4	60000004	06-MAR-2026	11-MAR-2026	1	08-JAN-2026	1500	CONFIRMATA	50004	40001
5	60000005	30-DEC-2025	02-JAN-2026	4	29-OCT-2025	2250	ANULATA	50005	40022
6	60000006	23-DEC-2025	27-DEC-2025	3	02-NOV-2025	2200	FINALIZATA	50002	40013
7	60000007	18-NOV-2025	21-NOV-2025	4	06-NOV-2025	2100	FINALIZATA	50003	40007

```
INSERT INTO facturi VALUES (70000001, 2500, TO_DATE('04-01-
2026', 'DD-MM-YYYY'), 'CARD', TO_DATE('04-01-2026', 'DD-MM-
YYYY'), 'PLATITA', 60000001);

INSERT INTO facturi VALUES (70000002, 1350, TO_DATE('20-12-
2025', 'DD-MM-YYYY'), NULL, NULL, 'NEPLATITA', 60000002);

INSERT INTO facturi VALUES (70000003, 2400, TO_DATE('20-07-
2025', 'DD-MM-YYYY'), 'CARD', TO_DATE('20-07-2025', 'DD-MM-
YYYY'), 'PLATITA', 60000003);

INSERT INTO facturi VALUES (70000004, 1500, TO_DATE('08-01-
2026', 'DD-MM-YYYY'), 'CARD', TO_DATE('08-01-2026', 'DD-MM-
YYYY'), 'PLATITA', 60000004);

INSERT INTO facturi VALUES (70000005, 2250, TO_DATE('29-10-
2025', 'DD-MM-YYYY'), NULL, NULL, 'NEPLATITA', 60000005);
```

```
INSERT INTO facturi VALUES (70000006, 2200, TO_DATE('02-11-2025', 'DD-MM-YYYY'), 'NUMERAR', TO_DATE('27-12-2025', 'DD-MM-YYYY'), 'PLATITA', 60000006);

INSERT INTO facturi VALUES (70000007, 2100, TO_DATE('06-11-2025', 'DD-MM-YYYY'), 'CARD', TO_DATE('06-11-2025', 'DD-MM-YYYY'), 'PLATITA', 60000007);

INSERT INTO facturi VALUES (70000008, 1250, TO_DATE('04-01-2026', 'DD-MM-YYYY'), 'CARD', TO_DATE('04-01-2026', 'DD-MM-YYYY'), 'PLATITA', 60000001);

INSERT INTO facturi VALUES (70000009, 200, TO_DATE('04-01-2026', 'DD-MM-YYYY'), 'CARD', TO_DATE('04-01-2026', 'DD-MM-YYYY'), 'PLATITA', 60000001);

INSERT INTO facturi VALUES (70000010, 180, TO_DATE('20-12-2025', 'DD-MM-YYYY'), NULL, NULL, 'NEPLATITA', 60000002);

INSERT INTO facturi VALUES (70000011, 150, TO_DATE('08-01-2026', 'DD-MM-YYYY'), 'CARD', TO_DATE('08-01-2026', 'DD-MM-YYYY'), 'PLATITA', 60000004);

INSERT INTO facturi VALUES (70000012, 0, TO_DATE('11-02-2026', 'DD-MM-YYYY'), NULL, TO_DATE('11-02-2026', 'DD-MM-YYYY'), 'PLATITA', 60000004);

INSERT INTO facturi VALUES (70000013, 480, TO_DATE('29-10-2025', 'DD-MM-YYYY'), NULL, NULL, 'NEPLATITA', 60000005);

INSERT INTO facturi VALUES (70000014, 60, TO_DATE('24-12-2025', 'DD-MM-YYYY'), 'NUMERAR', TO_DATE('27-12-2025', 'DD-MM-YYYY'), 'PLATITA', 60000006);

INSERT INTO facturi VALUES (70000015, 0, TO_DATE('02-11-2025', 'DD-MM-YYYY'), NULL, TO_DATE('02-11-2025', 'DD-MM-YYYY'), 'PLATITA', 60000006);
```

```
INSERT INTO facturi VALUES (70000016, 360, TO_DATE('06-11-2025',
'DD-MM-YYYY'), 'CARD', TO_DATE('06-11-2025', 'DD-MM-YYYY'),
'PLATITA', 60000007);

INSERT INTO facturi VALUES (70000017, 0, TO_DATE('06-11-2025',
'DD-MM-YYYY'), NULL, TO_DATE('06-11-2025', 'DD-MM-YYYY'),
'PLATITA', 60000007);
```



FACTURI

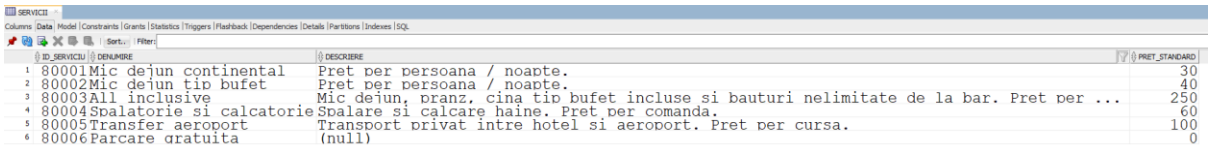
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter:

	ID_FACTURA	SUMA	DATA_EMITERE	METODA_PLATA	DATA_PLATA	STARE_PLATA	ID_REZERVARE
1	70000001	2500	04-JAN-2026	CARD	04-JAN-2026	PLATITA	60000001
2	70000002	1350	20-DEC-2025	(null)	(null)	NEPLATITA	60000002
3	70000003	2400	20-JUL-2025	CARD	20-JUL-2025	PLATITA	60000003
4	70000004	1500	08-JAN-2026	CARD	08-JAN-2026	PLATITA	60000004
5	70000005	2250	29-OCT-2025	(null)	(null)	NEPLATITA	60000005
6	70000006	2200	02-NOV-2025	NUMERAR	27-DEC-2025	PLATITA	60000006
7	70000007	2100	06-NOV-2025	CARD	06-NOV-2025	PLATITA	60000007
8	70000008	1250	04-JAN-2026	CARD	04-JAN-2026	PLATITA	60000001
9	70000009	200	04-JAN-2026	CARD	04-JAN-2026	PLATITA	60000001
10	70000010	180	20-DEC-2025	(null)	(null)	NEPLATITA	60000002
11	70000011	150	08-JAN-2026	CARD	08-JAN-2026	PLATITA	60000004
12	70000012	0	11-FEB-2026	(null)	11-FEB-2026	PLATITA	60000004
13	70000013	480	29-OCT-2025	(null)	(null)	NEPLATITA	60000005
14	70000014	60	24-DEC-2025	NUMERAR	27-DEC-2025	PLATITA	60000006
15	70000015	0	02-NOV-2025	(null)	02-NOV-2025	PLATITA	60000006
16	70000016	360	06-NOV-2025	CARD	06-NOV-2025	PLATITA	60000007
17	70000017	0	06-NOV-2025	(null)	06-NOV-2025	PLATITA	60000007

```
INSERT INTO servicii VALUES(80001, 'Mic dejun continental',
'Pret per persoana / noapte.', 30);

INSERT INTO servicii VALUES(80002, 'Mic dejun tip bufet', 'Pret
per persoana / noapte.', 40);

INSERT INTO servicii VALUES(80003, 'All inclusive', 'Mic dejun,
pranz, cina tip bufet incluse si bauturi nelimitate de la bar.
Pret per persoana / noapte', 250);

INSERT INTO servicii VALUES(80004, 'Spalatorie si calcatorie',
'Spalare si calcare haine. Pret per comanda.', 60);

INSERT INTO servicii VALUES(80005, 'Transfer aeroport',
'Transport privat intre hotel si aeroport. Pret per cursa.',
100);
```

```
INSERT INTO servicii VALUES(80006, 'Parcare gratuita', NULL, 0);
```



	ID_SERVICIU	DENUMIRE	DESCRIERE	PRET_STANDARD
1	80001	Mic dejun continental	Pret per persoana / noapte.	30
2	80002	Mic dejun tip bufet	Pret per persoana / noapte.	40
3	80003	All inclusive	Mic dejun, pranz, cina tip bufet incluse si bauturi nelimitate de la bar. Pret per ...	250
4	80004	Spalatorie si calcatorie	Spalare si calcare haine. Pret per comanda.	60
5	80005	Transfer aeroport	Transport privat intre hotel si aeroport. Pret per cursa.	100
6	80006	Parcare gratuita	(null)	0

```
INSERT INTO servicii_rezervari VALUES(60000001, 80003, 5, 1250);

INSERT INTO servicii_rezervari VALUES(60000001, 80005, 2, 200);

INSERT INTO servicii_rezervari VALUES(60000002, 80001, 6, 180);

INSERT INTO servicii_rezervari VALUES(60000004, 80001, 5, 150);

INSERT INTO servicii_rezervari VALUES(60000004, 80006, 1, 0);

INSERT INTO servicii_rezervari VALUES(60000005, 80002, 12, 480);

INSERT INTO servicii_rezervari VALUES(60000006, 80004, 1, 60);

INSERT INTO servicii_rezervari VALUES(60000006, 80006, 1, 0);

INSERT INTO servicii_rezervari VALUES(60000007, 80001, 12, 360);

INSERT INTO servicii_rezervari VALUES(60000007, 80006, 1, 0);
```

SERVICII_REZERVARI

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependenc

Sort.. | Filter:

	ID_REZERVARE	ID_SERVICIU	CANTITATE	PRET_APLICAT
1	60000001	80003	5	1250
2	60000001	80005	2	200
3	60000002	80001	6	180
4	60000004	80001	5	150
5	60000004	80006	1	0
6	60000005	80002	12	480
7	60000006	80004	1	60
8	60000006	80006	1	0
9	60000007	80001	12	360
10	60000007	80006	1	0

```
COMMIT;
```

Cerința 6

Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.

Pentru un **interval de timp** dat, să se afișeze **hotelul cu cele mai mari încasări**, dar acceptând doar **anumite metode de plată**. Să se afișeze și **numărul total de facturi plătite la toate hotelurile** în perioada respectivă, **suma** totală și **media** acestora.

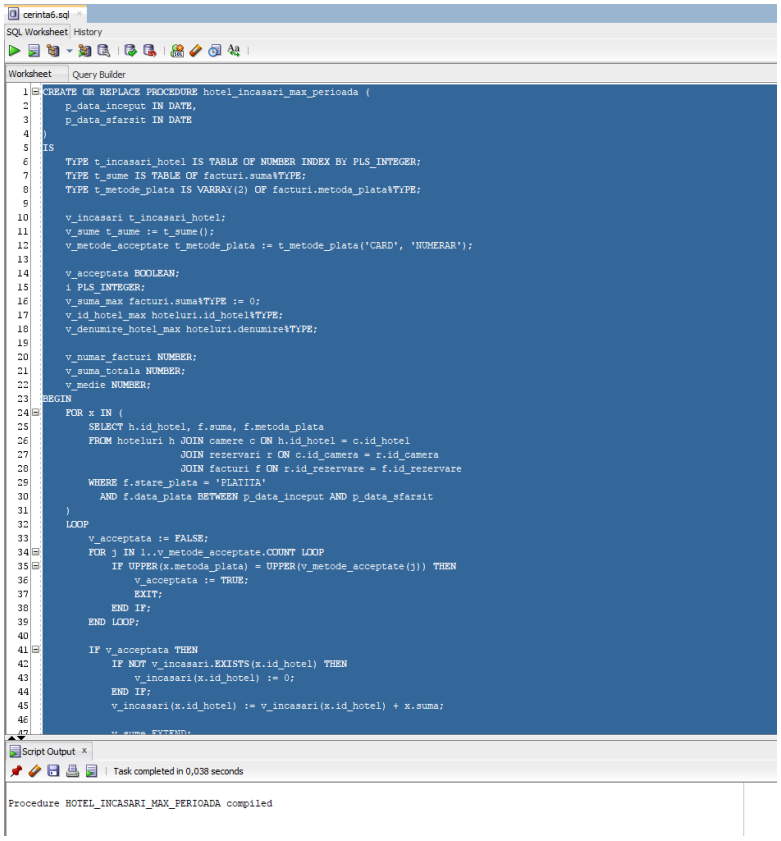
```
CREATE OR REPLACE PROCEDURE hotel_incasari_max_perioada (
    p_data_inceput IN DATE,
    p_data_sfarsit IN DATE
)
IS
    TYPE t_incasari_hotel IS TABLE OF NUMBER INDEX BY
PLS_INTEGER;
    TYPE t_sume IS TABLE OF facturi.suma%TYPE;
    TYPE t_metode_plata IS VARRAY(2) OF
facturi.metoda_plata%TYPE;

    v_incasari t_incasari_hotel;
    v_sume t_sume := t_sume();
    v_metode_acceptate t_metode_plata :=
t_metode_plata('CARD', 'NUMERAR');

    v_acceptata BOOLEAN;
    i PLS_INTEGER;
```

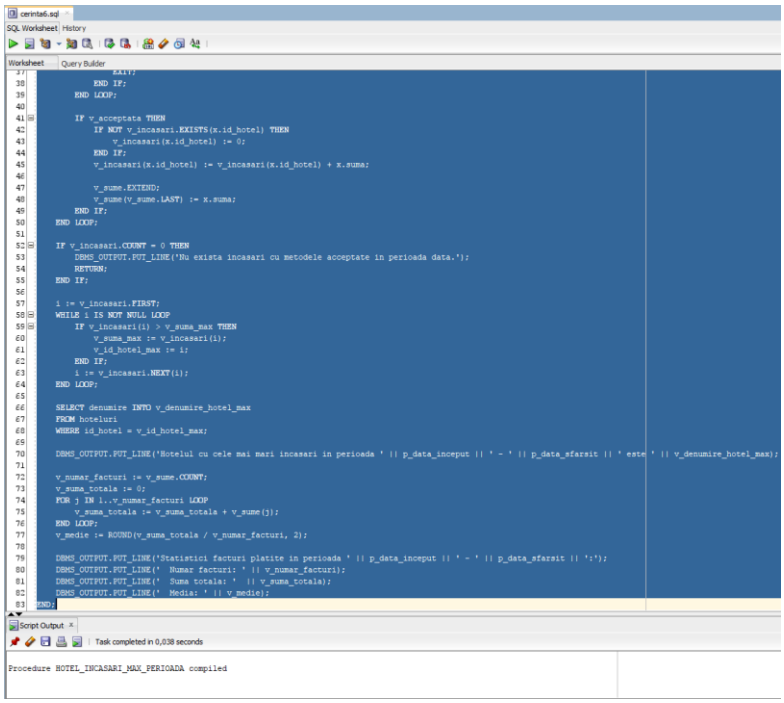
```
    v_suma_max facturi.suma%TYPE := 0;
    v_id_hotel_max hoteluri.id_hotel%TYPE;
    v_denumire_hotel_max hoteluri.denumire%TYPE;

    v_numar_facturi NUMBER;
    v_suma_totala NUMBER;
    v_medie NUMBER;
BEGIN
    FOR x IN (
        SELECT h.id_hotel, f.suma, f.metoda_plata
        FROM hoteluri h JOIN camere c ON h.id_hotel =
c.id_hotel
                        JOIN rezervari r ON c.id_camera =
r.id_camera
                        JOIN facturi f ON r.id_rezervare =
f.id_rezervare
        WHERE f.stare_plata = 'PLATITA'
          AND f.data_plata BETWEEN p_data_inceput AND
p_data_sfarsit
    )
    LOOP
        v_acceptata := FALSE;
        FOR j IN 1..v_metode_acceptate.COUNT LOOP
            IF UPPER(x.metoda_plata) =
UPPER(v_metode_acceptate(j)) THEN
                v_acceptata := TRUE;
```

```
                EXIT;
            END IF;
        END LOOP;

        IF v_acceptata THEN
            IF NOT v_incasari.EXISTS(x.id_hotel) THEN
                v_incasari(x.id_hotel) := 0;
            END IF;
            v_incasari(x.id_hotel) := v_incasari(x.id_hotel) +
x.suma;

            v_sume.EXTEND;
            v_sume(v_sume.LAST) := x.suma;
        END IF;
    END LOOP;

    IF v_incasari.COUNT = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista incasari cu metodele
acceptate in perioada data.');
        RETURN;
    END IF;

    i := v_incasari.FIRST;
    WHILE i IS NOT NULL LOOP
        IF v_incasari(i) > v_suma_max THEN
```

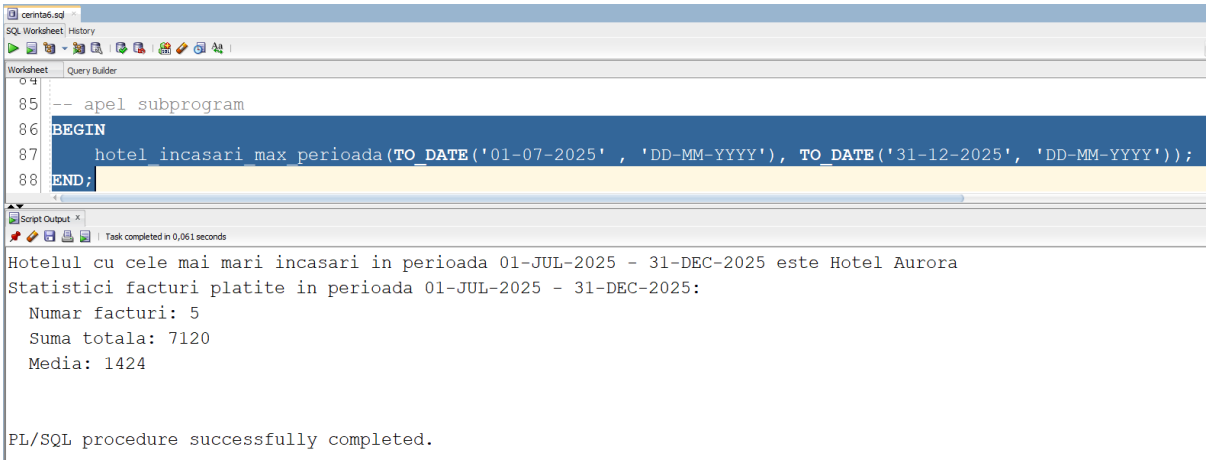
```
            v_suma_max := v_incasari(i);
            v_id_hotel_max := i;
        END IF;
        i := v_incasari.NEXT(i);
    END LOOP;

    SELECT denumire INTO v_denumire_hotel_max
    FROM hoteluri
    WHERE id_hotel = v_id_hotel_max;

    DBMS_OUTPUT.PUT_LINE('Hotelul cu cele mai mari incasari in
perioada ' || p_data_inceput || ' - ' || p_data_sfarsit || '
este ' || v_denumire_hotel_max);

    v_numar_facturi := v_sume.COUNT;
    v_suma_totala := 0;
    FOR j IN 1..v_numar_facturi LOOP
        v_suma_totala := v_suma_totala + v_sume(j);
    END LOOP;
    v_medie := ROUND(v_suma_totala / v_numar_facturi, 2);

    DBMS_OUTPUT.PUT_LINE('Statistici facturi platite in
perioada ' || p_data_inceput || ' - ' || p_data_sfarsit ||
':');
    DBMS_OUTPUT.PUT_LINE('  Numar facturi: ' ||
v_numar_facturi);
```

```
    DBMS_OUTPUT.PUT_LINE('  Suma totala: '  || v_suma_totala);
    DBMS_OUTPUT.PUT_LINE('  Media: ' || v_medie);
END;
/
```





```
-- apel subprogram

BEGIN

    hotel_incasari_max_perioada(TO_DATE('01-07-2025' , 'DD-MM-
YYYY'), TO_DATE('31-12-2025', 'DD-MM-YYYY'));

END;

/
```



Cerința 7

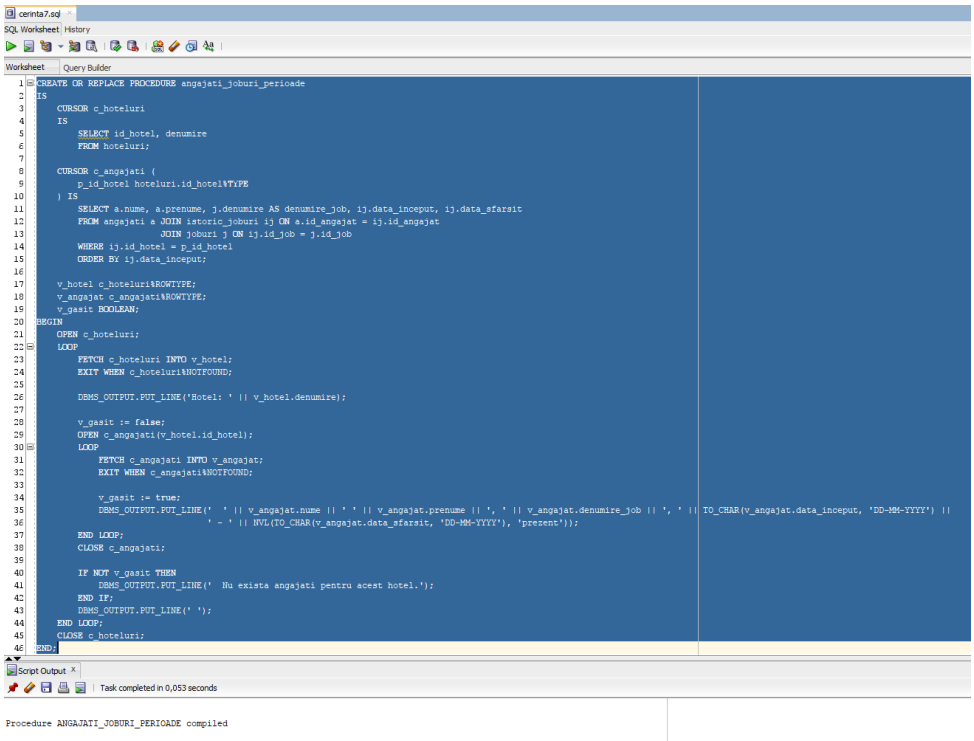
Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogramul.

Să se afișeze **pentru fiecare hotel**, **angajații** care au lucrat în acel hotel, **joburile** pe care le-au ocupat și **perioadele**, în ordine crescătoare a datelor la care au început să lucreze.

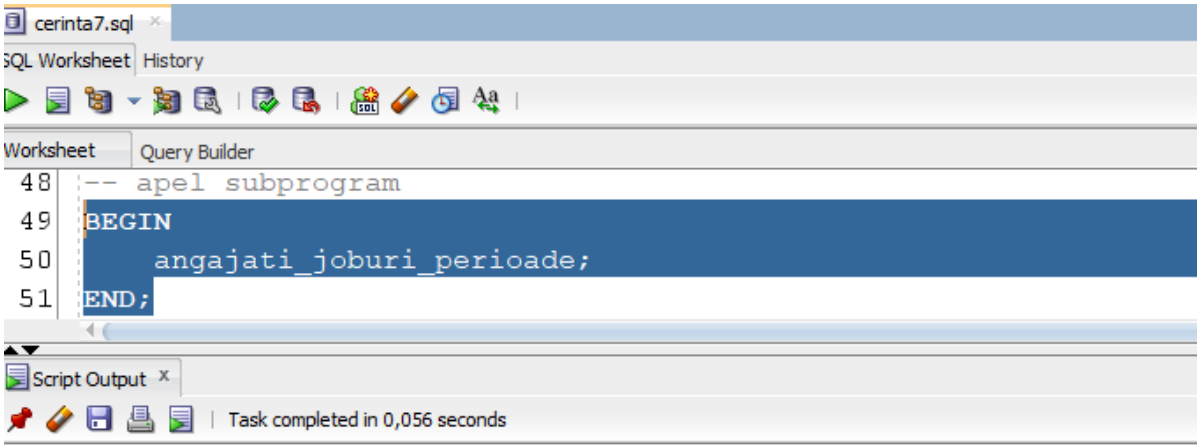
```
CREATE OR REPLACE PROCEDURE angajati_joburi_perioade
IS
    CURSOR c_hoteluri
    IS
        SELECT id_hotel, denumire
        FROM hoteluri;

    CURSOR c_angajati (
        p_id_hotel hoteluri.id_hotel%TYPE
    ) IS
        SELECT a.nume, a.prenume, j.denumire AS denumire_job,
ij.data_inceput, ij.data_sfarsit
        FROM angajati a JOIN istoric_joburi ij ON a.id_angajat
= ij.id_angajat
                        JOIN joburi j ON ij.id_job = j.id_job
        WHERE ij.id_hotel = p_id_hotel
        ORDER BY ij.data_inceput;
```

```
    v_hotel c_hoteluri%ROWTYPE;
    v_angajat c_angajati%ROWTYPE;
    v_gasit BOOLEAN;
BEGIN
    OPEN c_hoteluri;
    LOOP
        FETCH c_hoteluri INTO v_hotel;
        EXIT WHEN c_hoteluri%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('Hotel: ' || v_hotel.denumire);

        v_gasit := false;
        OPEN c_angajati(v_hotel.id_hotel);
        LOOP
            FETCH c_angajati INTO v_angajat;
            EXIT WHEN c_angajati%NOTFOUND;

            v_gasit := true;
            DBMS_OUTPUT.PUT_LINE('  ' || v_angajat.nume || ' '
|| v_angajat.prenume || ', ' || v_angajat.denumire_job || ', '
|| TO_CHAR(v_angajat.data_inceput, 'DD-MM-YYYY') ||
                                ' - ' ||
NVL(TO_CHAR(v_angajat.data_sfarsit, 'DD-MM-YYYY'),
'prezent'));
        END LOOP;
        CLOSE c_angajati;
```

```
        IF NOT v_gasit THEN
            DBMS_OUTPUT.PUT_LINE('  Nu exista angajati pentru
acest hotel.');
        END IF;
        DBMS_OUTPUT.PUT_LINE(' ');
    END LOOP;
    CLOSE c_hoteluri;
END;
/
```




```
-- apel subprogram
BEGIN
    angajati_joburi_perioade;
END;
/
```




```
Hotel: Hotel Central
  Dumitru Andrei, Receptioner, 30-04-2015 - 14-07-2023
  Popescu Ion, Asistent manager, 25-02-2016 - 18-01-2017
  Popescu Ion, Manager, 19-01-2017 - prezent
  Dumitru Andrei, Administrator IT, 03-11-2019 - prezent
  Ionescu Ana, Receptioner, 15-07-2023 - prezent
  Lazar Alexandru, Camerista, 21-08-2024 - prezent

Hotel: Hotel Aurora
  Grigore Elena, Camerista, 25-09-2021 - prezent
  Barbu Mihai, Receptioner, 01-10-2021 - prezent
  Lazar Alexandru, Manager, 28-05-2022 - prezent

Hotel: Hotel Metropolis
  Costache Teodor, Bucatar, 16-01-2018 - prezent
  Vasilescu Maria, Manager, 20-06-2019 - prezent
  Grigore Elena, Camerista, 12-03-2020 - 24-09-2021
  Negru Andreea, Receptioner, 15-10-2020 - prezent
  Dumitrescu Bogdan, Receptioner, 08-03-2021 - prezent
  Radulescu Lucian, Tehnician mentenanta, 18-03-2021 - 04-08-2023
  Preda Alexandra, Camerista, 25-09-2021 - prezent
  Serban Ionel, Barman, 02-04-2022 - prezent
  Munteanu Simona, Tehnician mentenanta, 05-08-2023 - prezent
  Badea Sebastian, Barman, 12-06-2024 - prezent

Hotel: Hotel Solaris
  Nu exista angajati pentru acest hotel.

Hotel: Hotel Diamant
  Nu exista angajati pentru acest hotel.



PL/SQL procedure successfully completed.
```

Cerința 8

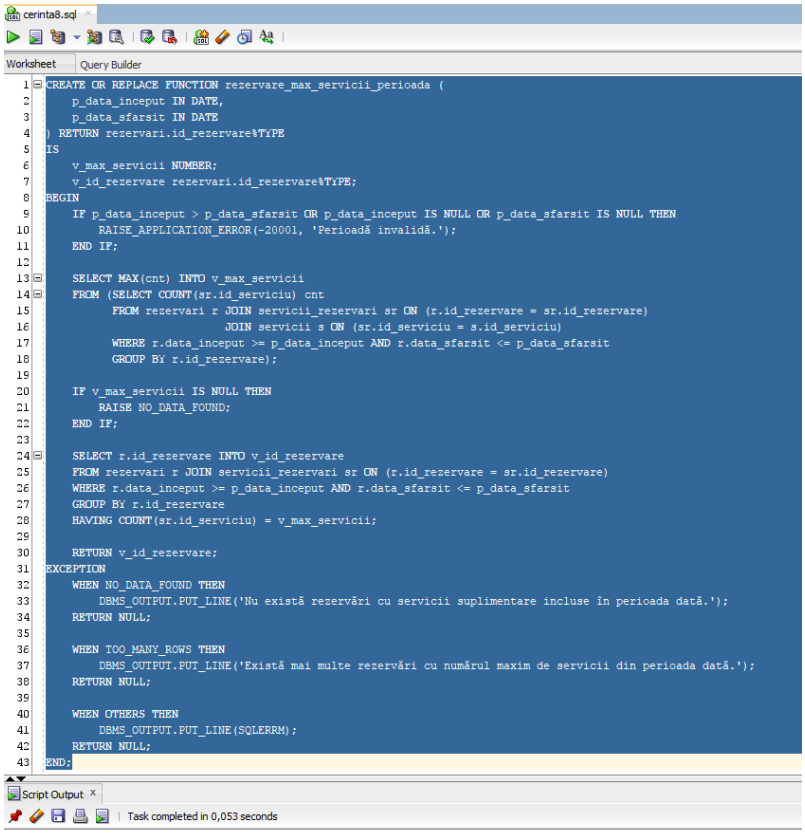
Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele create. Tratați toate excepțiile care pot apărea, incluzând excepțiile predefinite NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Se dă un **interval de timp** (data de început și data de sfârșit). Să se găsească **rezervarea** inclusă în acest interval care are **cele mai multe servicii** suplimentare incluse.

```
CREATE OR REPLACE FUNCTION rezervare_max_servicii_perioada (
    p_data_inceput IN DATE,
    p_data_sfarsit IN DATE
) RETURN rezervari.id_rezervare%TYPE
IS
    v_max_servicii NUMBER;
    v_id_rezervare rezervari.id_rezervare%TYPE;
BEGIN
    IF p_data_inceput > p_data_sfarsit OR p_data_inceput IS
NULL OR p_data_sfarsit IS NULL THEN
        RAISE_APPLICATION_ERROR(-20001, 'Perioadă invalidă.');
    END IF;

    SELECT MAX(cnt) INTO v_max_servicii
    FROM (SELECT COUNT(sr.id_serviciu) cnt
          FROM rezervari r JOIN servicii_rezervari sr ON
(r.id_rezervare = sr.id_rezervare)
```

```
                        JOIN servicii s ON (sr.id_serviciu
= s.id_serviciu)
          WHERE r.data_inceput >= p_data_inceput AND
r.data_sfarsit <= p_data_sfarsit
          GROUP BY r.id_rezervare);

    IF v_max_servicii IS NULL THEN
        RAISE NO_DATA_FOUND;
    END IF;

    SELECT r.id_rezervare INTO v_id_rezervare
    FROM rezervari r JOIN servicii_rezervari sr ON
(r.id_rezervare = sr.id_rezervare)
    WHERE r.data_inceput >= p_data_inceput AND r.data_sfarsit
<= p_data_sfarsit
    GROUP BY r.id_rezervare
    HAVING COUNT(sr.id_serviciu) = v_max_servicii;

    RETURN v_id_rezervare;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nu există rezervări cu servicii
suplimentare incluse în perioada dată.');
    RETURN NULL;

    WHEN TOO_MANY_ROWS THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Există mai multe rezervări cu
numărul maxim de servicii din perioada dată.');
    RETURN NULL;


    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
    RETURN NULL;
END;
/
```

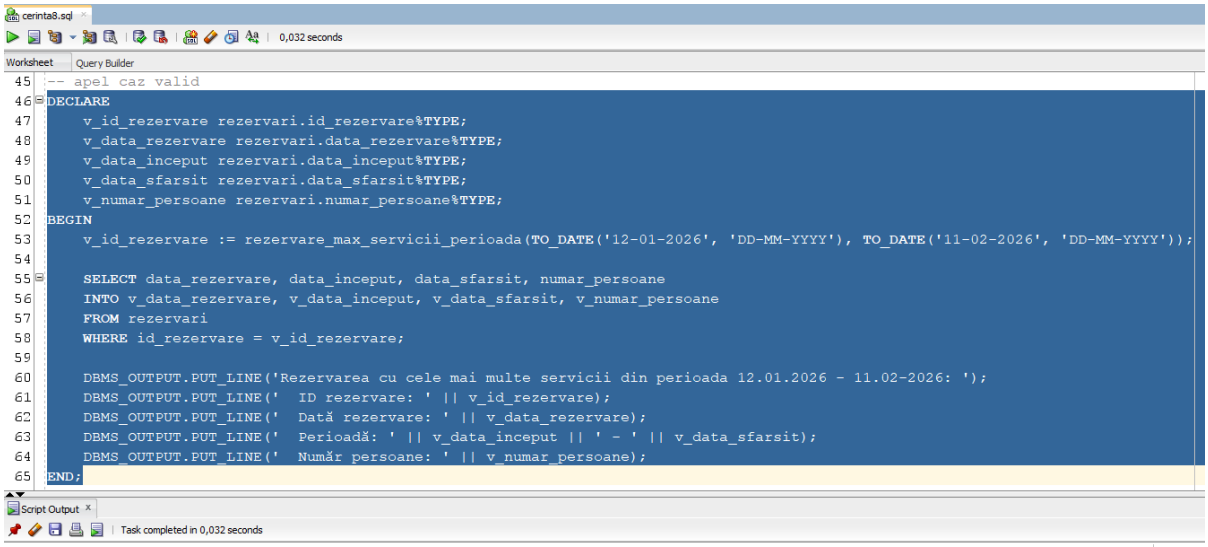



```
CREATE OR REPLACE FUNCTION rezervare_max_servicii_perioada (
    p_data_inceput IN DATE,
    p_data_sfarsit IN DATE
) RETURN rezervari.id_rezervare%TYPE
IS
    v_max_servicii NUMBER;
    v_id_rezervare rezervari.id_rezervare%TYPE;
BEGIN
    IF p_data_inceput > p_data_sfarsit OR p_data_inceput IS NULL OR p_data_sfarsit IS NULL THEN
        RAISE_APPLICATION_ERROR(-20001, 'Perioadă invalidă.');
    END IF;

    SELECT MAX(cnt) INTO v_max_servicii
    FROM (SELECT COUNT(sr.id_serviciu) cnt
          FROM rezervari r JOIN servicii_rezervari sr ON (r.id_rezervare = sr.id_rezervare)
                           JOIN servicii s ON (sr.id_serviciu = s.id_serviciu)
          WHERE r.data_inceput >= p_data_inceput AND r.data_sfarsit <= p_data_sfarsit
          GROUP BY r.id_rezervare);

    IF v_max_servicii IS NULL THEN
        RAISE NO_DATA_FOUND;
    END IF;

    SELECT r.id_rezervare INTO v_id_rezervare
    FROM rezervari r JOIN servicii_rezervari sr ON (r.id_rezervare = sr.id_rezervare)
    WHERE r.data_inceput >= p_data_inceput AND r.data_sfarsit <= p_data_sfarsit
    GROUP BY r.id_rezervare
    HAVING COUNT(sr.id_serviciu) = v_max_servicii;

    RETURN v_id_rezervare;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nu există rezervări cu servicii suplimentare incluse în perioada dată.');
    RETURN NULL;

    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Există mai multe rezervări cu numărul maxim de servicii din perioada dată.');
    RETURN NULL;

    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
    RETURN NULL;
END;
```

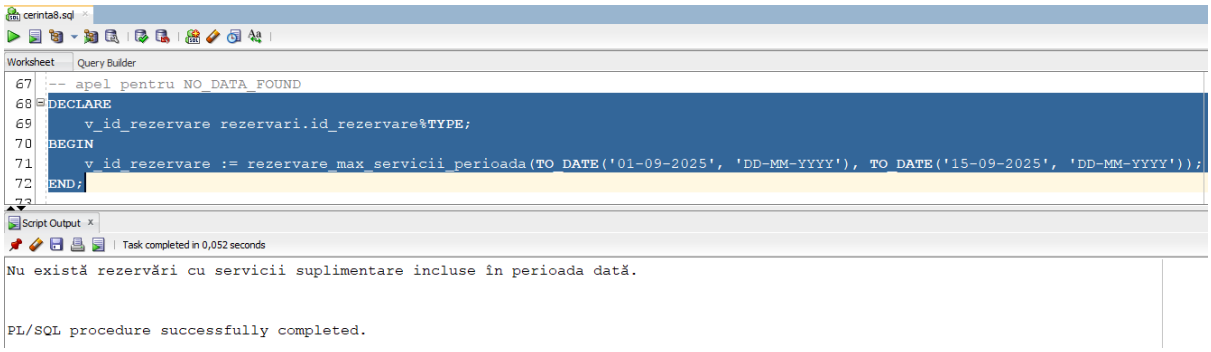
```
Function REZERVARE_MAX_SERVICII_PERIOADA compiled
```

```
-- apel caz valid
DECLARE
    v_id_rezervare rezervari.id_rezervare%TYPE;
    v_data_rezervare rezervari.data_rezervare%TYPE;
```

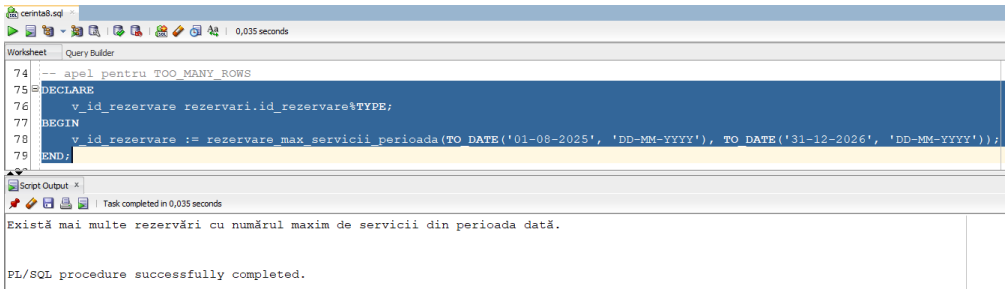
```
    v_data_inceput rezervari.data_inceput%TYPE;
    v_data_sfarsit rezervari.data_sfarsit%TYPE;
    v_numar_persoane rezervari.numar_persoane%TYPE;
BEGIN
    v_id_rezervare :=
rezervare_max_servicii_perioada(TO_DATE('12-01-2026', 'DD-MM-
YYYY'), TO_DATE('11-02-2026', 'DD-MM-YYYY'));

    SELECT data_rezervare, data_inceput, data_sfarsit,
numar_persoane
    INTO v_data_rezervare, v_data_inceput, v_data_sfarsit,
v_numar_persoane
    FROM rezervari
    WHERE id_rezervare = v_id_rezervare;

    DBMS_OUTPUT.PUT_LINE('Rezervarea cu cele mai multe
servicii din perioada 12.01.2026 - 11.02-2026: ');
    DBMS_OUTPUT.PUT_LINE('  ID rezervare: ' ||
v_id_rezervare);
    DBMS_OUTPUT.PUT_LINE('  Dată rezervare: ' ||
v_data_rezervare);
    DBMS_OUTPUT.PUT_LINE('  Perioadă: ' || v_data_inceput || '
- ' || v_data_sfarsit);
    DBMS_OUTPUT.PUT_LINE('  Număr persoane: ' ||
v_numar_persoane);
END;
/
```



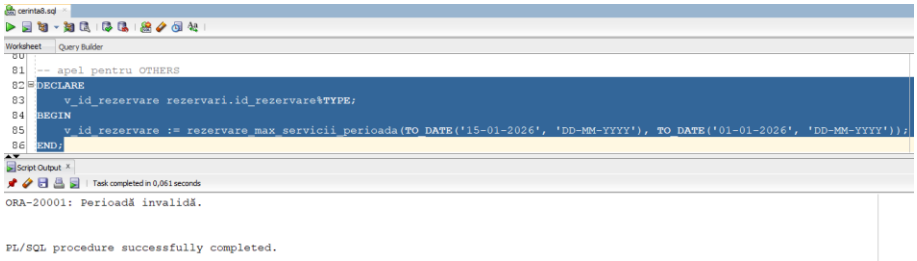

```
-- apel pentru NO_DATA_FOUND

DECLARE

    v_id_rezervare rezervari.id_rezervare%TYPE;

BEGIN

    v_id_rezervare :=
rezervare_max_servicii_perioada(TO_DATE('01-09-2025', 'DD-MM-
YYYY'), TO_DATE('15-09-2025', 'DD-MM-YYYY'));

END;

/
```



```
-- apel pentru TOO_MANY_ROWS
DECLARE
    v_id_rezervare rezervari.id_rezervare%TYPE;
BEGIN
    v_id_rezervare :=
rezervare_max_servicii_perioada(TO_DATE('01-08-2025', 'DD-MM-
YYYY'), TO_DATE('31-12-2026', 'DD-MM-YYYY'));
END;
/
```




```
-- apel pentru OTHERS
DECLARE
    v_id_rezervare rezervari.id_rezervare%TYPE;
BEGIN
    v_id_rezervare :=
rezervare_max_servicii_perioada(TO_DATE('15-01-2026', 'DD-MM-
YYYY'), TO_DATE('01-01-2026', 'DD-MM-YYYY'));
END;
/
```



Cerința 9

Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să aibă minim 2 parametri și să utilizeze într-o singură comandă SQL 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel de sistem. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.

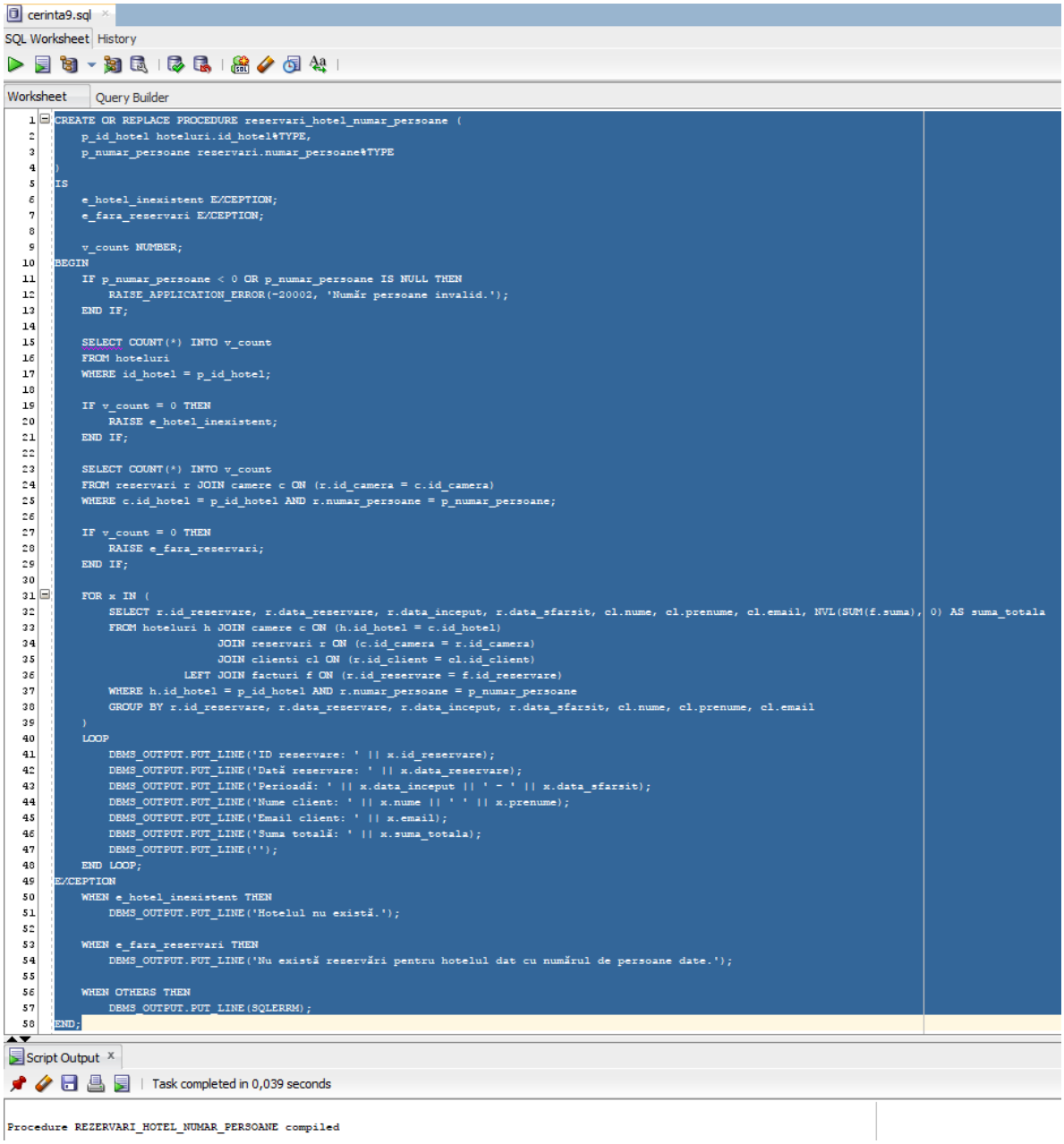
Se dau un **hotel** și un **număr de persoane**. Să se afișeze informații despre **rezervare, client și suma totală de plată** (suma tuturor facturilor asociate unei rezervări) pentru **toate rezervările** de la acel hotel pentru numărul dat de persoane.

```
CREATE OR REPLACE PROCEDURE rezervari_hotel_numar_persoane (
    p_id_hotel hoteluri.id_hotel%TYPE,
    p_numar_persoane rezervari.numar_persoane%TYPE
)
IS
    e_hotel_inexistent EXCEPTION;
    e_fara_rezervari EXCEPTION;

    v_count NUMBER;
BEGIN
    IF p_numar_persoane < 0 OR p_numar_persoane IS NULL THEN
        RAISE_APPLICATION_ERROR(-20002, 'Număr persoane
invalid.');
    END IF;
```

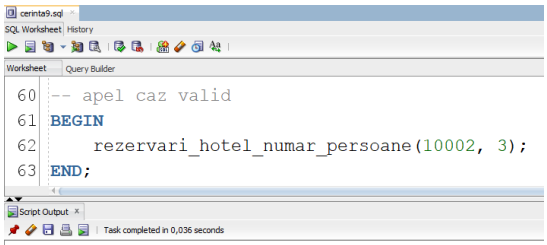
```
    SELECT COUNT(*) INTO v_count
    FROM hoteluri
    WHERE id_hotel = p_id_hotel;

    IF v_count = 0 THEN
        RAISE e_hotel_inexistent;
    END IF;

    SELECT COUNT(*) INTO v_count
    FROM rezervari r JOIN camere c ON (r.id_camera =
c.id_camera)
    WHERE c.id_hotel = p_id_hotel AND r.numar_persoane =
p_numar_persoane;

    IF v_count = 0 THEN
        RAISE e_fara_rezervari;
    END IF;

    FOR x IN (
        SELECT r.id_rezervare, r.data_rezervare,
r.data_inceput, r.data_sfarsit, cl.nume, cl.prenume, cl.email,
NVL(SUM(f.suma), 0) AS suma_totala
        FROM hoteluri h JOIN camere c ON (h.id_hotel =
c.id_hotel)
                        JOIN rezervari r ON (c.id_camera =
r.id_camera)
```

```
                        JOIN clienti cl ON (r.id_client =
cl.id_client)
                    LEFT JOIN facturi f ON (r.id_rezervare =
f.id_rezervare)
        WHERE h.id_hotel = p_id_hotel AND r.numar_persoane =
p_numar_persoane
        GROUP BY r.id_rezervare, r.data_rezervare,
r.data_inceput, r.data_sfarsit, cl.nume, cl.prenume, cl.email
    )
    LOOP
        DBMS_OUTPUT.PUT_LINE('ID rezervare: ' ||
x.id_rezervare);
        DBMS_OUTPUT.PUT_LINE('Dată rezervare: ' ||
x.data_rezervare);
        DBMS_OUTPUT.PUT_LINE('Perioadă: ' || x.data_inceput ||
' - ' || x.data_sfarsit);
        DBMS_OUTPUT.PUT_LINE('Nume client: ' || x.nume || ' '
|| x.prenume);
        DBMS_OUTPUT.PUT_LINE('Email client: ' || x.email);
        DBMS_OUTPUT.PUT_LINE('Suma totală: ' ||
x.suma_totala);
        DBMS_OUTPUT.PUT_LINE('');
    END LOOP;
EXCEPTION
    WHEN e_hotel_inexistent THEN
        DBMS_OUTPUT.PUT_LINE('Hotelul nu există.');
```

```
    WHEN e_fara_rezervari THEN
        DBMS_OUTPUT.PUT_LINE('Nu există rezervări pentru hotelul dat cu numărul de persoane date.');

    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

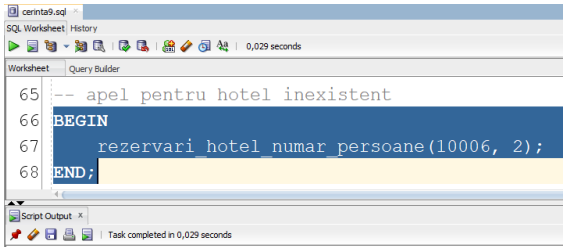


```
-- apel caz valid

BEGIN

    rezervari_hotel_numar_persoane(10002, 3);

END;

/
```



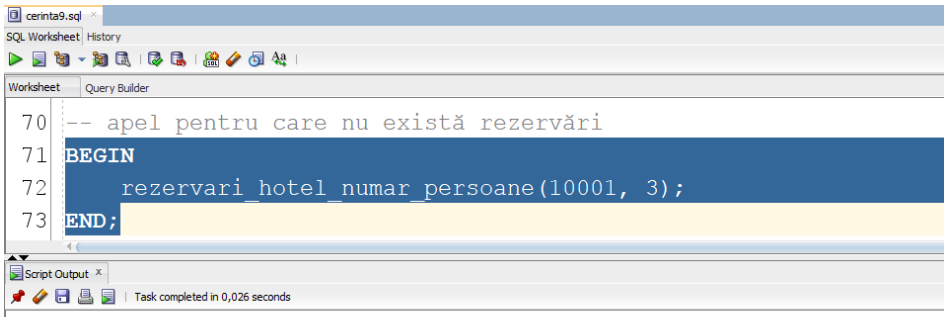

```
ID rezervare: 60000003
Dată rezervare: 20-JUL-2025
Perioadă: 01-AUG-2025 - 05-AUG-2025
Nume client: Ilie Florin
Email client: florin.ilie@yahoo.com
Suma totală: 2400

ID rezervare: 60000006
Dată rezervare: 02-NOV-2025
Perioadă: 23-DEC-2025 - 27-DEC-2025
Nume client: Marin Vlad
Email client: vlad.marin@gmail.com
Suma totală: 2260


PL/SQL procedure successfully completed.
```

```
-- apel pentru hotel inexistent

BEGIN

    rezervari_hotel_numar_persoane(10006, 2);

END;

/
```

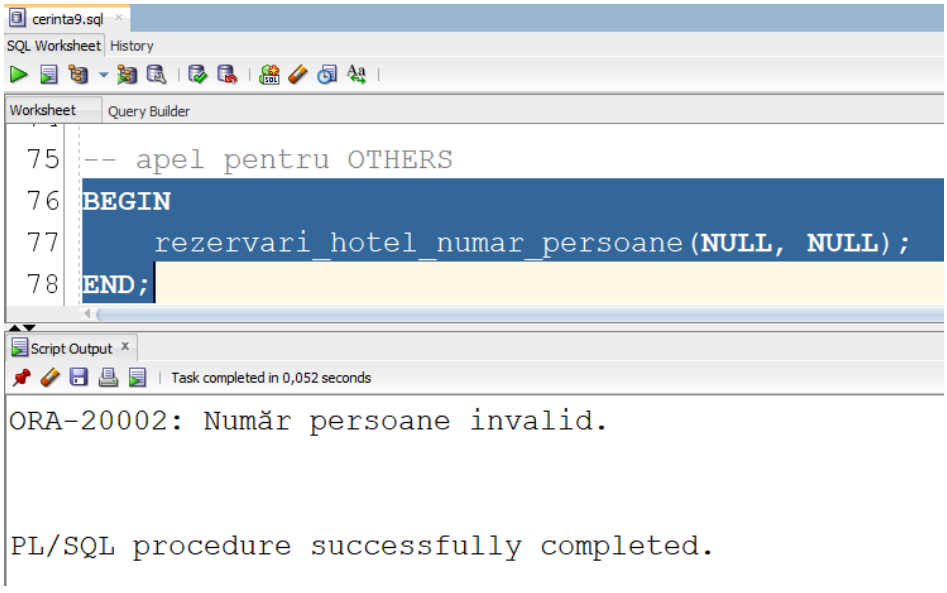



```
Hotelul nu există.


PL/SQL procedure successfully completed.
```

```
-- apel pentru care nu există rezervări

BEGIN

    rezervari_hotel_numar_persoane(10001, 3);

END;

/
```




```
-- apel pentru OTHERS

BEGIN

    rezervari_hotel_numar_persoane(NULL, NULL);

END;

/
```

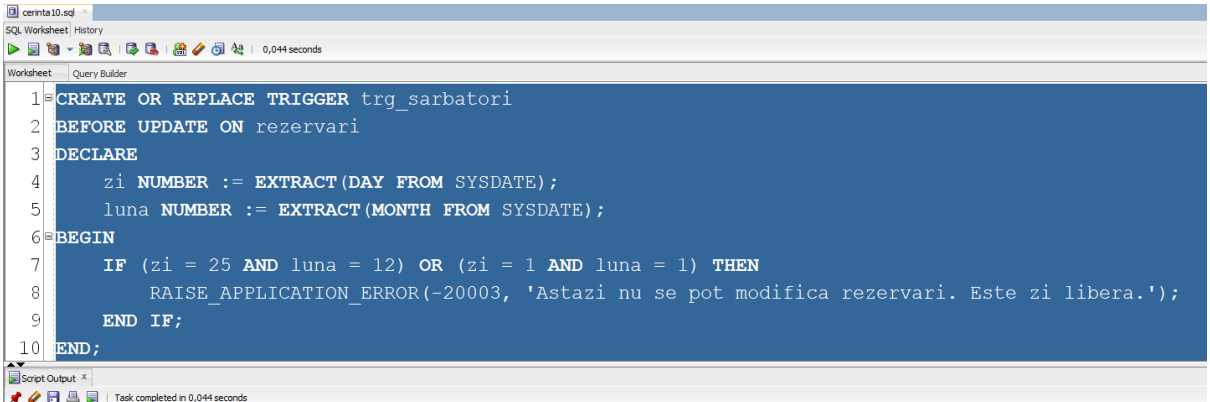


Cerința 10

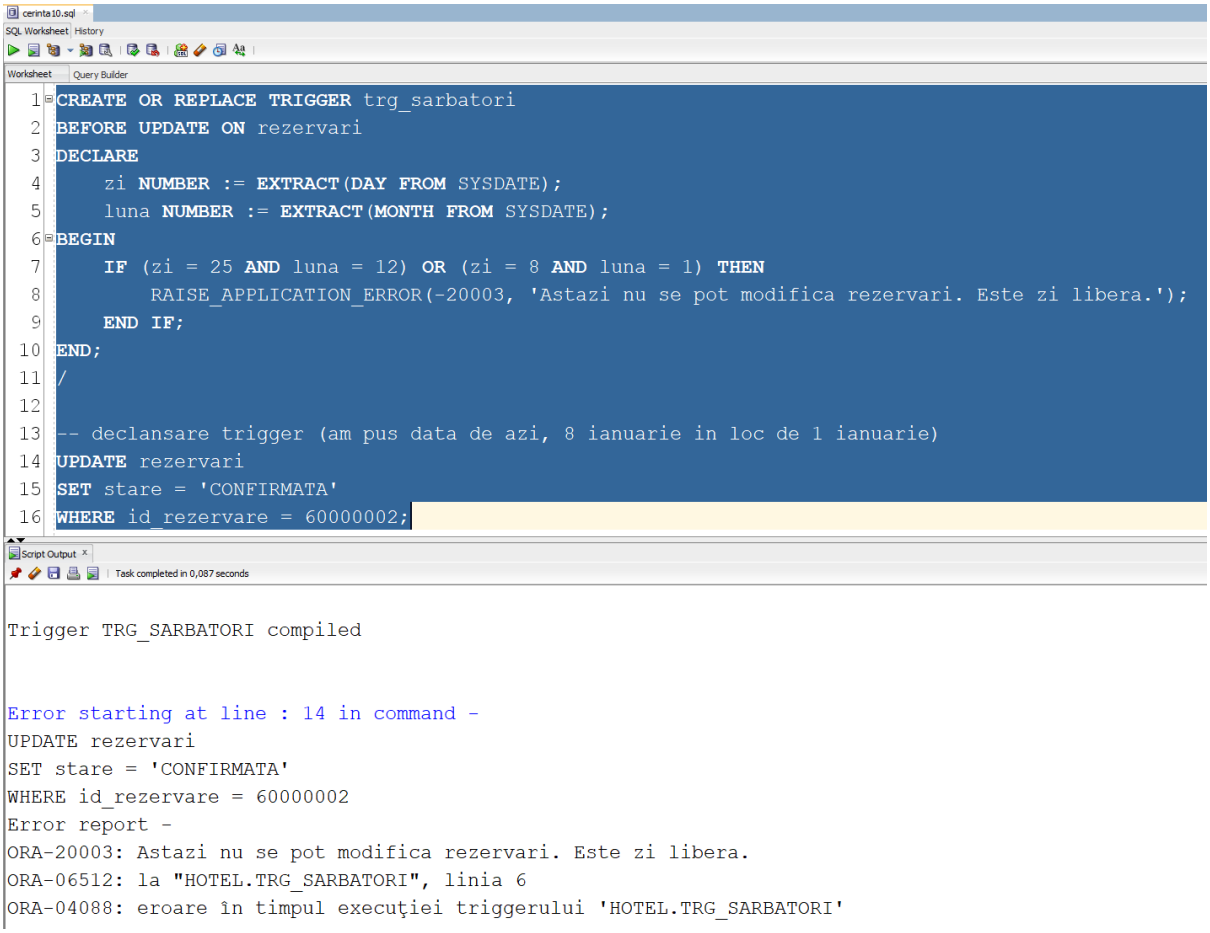
Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.

Să se creeze un trigger care **nu permite modificarea rezervărilor** pe 25 decembrie și 1 ianuarie.

```
CREATE OR REPLACE TRIGGER trg_sarbatori
BEFORE UPDATE ON rezervari
DECLARE
    zi NUMBER := EXTRACT(DAY FROM SYSDATE);
    luna NUMBER := EXTRACT(MONTH FROM SYSDATE);
BEGIN
    IF (zi = 25 AND luna = 12) OR (zi = 1 AND luna = 1) THEN
        RAISE_APPLICATION_ERROR(-20003, 'Astazi nu se pot
modifica rezervari. Este zi libera.');
    END IF;
END;
/
```



```
-- declansare trigger (am pus data de azi, 8 ianuarie in loc de 1 ianuarie)

UPDATE rezervari

SET stare = 'CONFIRMATA'

WHERE id_rezervare = 60000002;
```



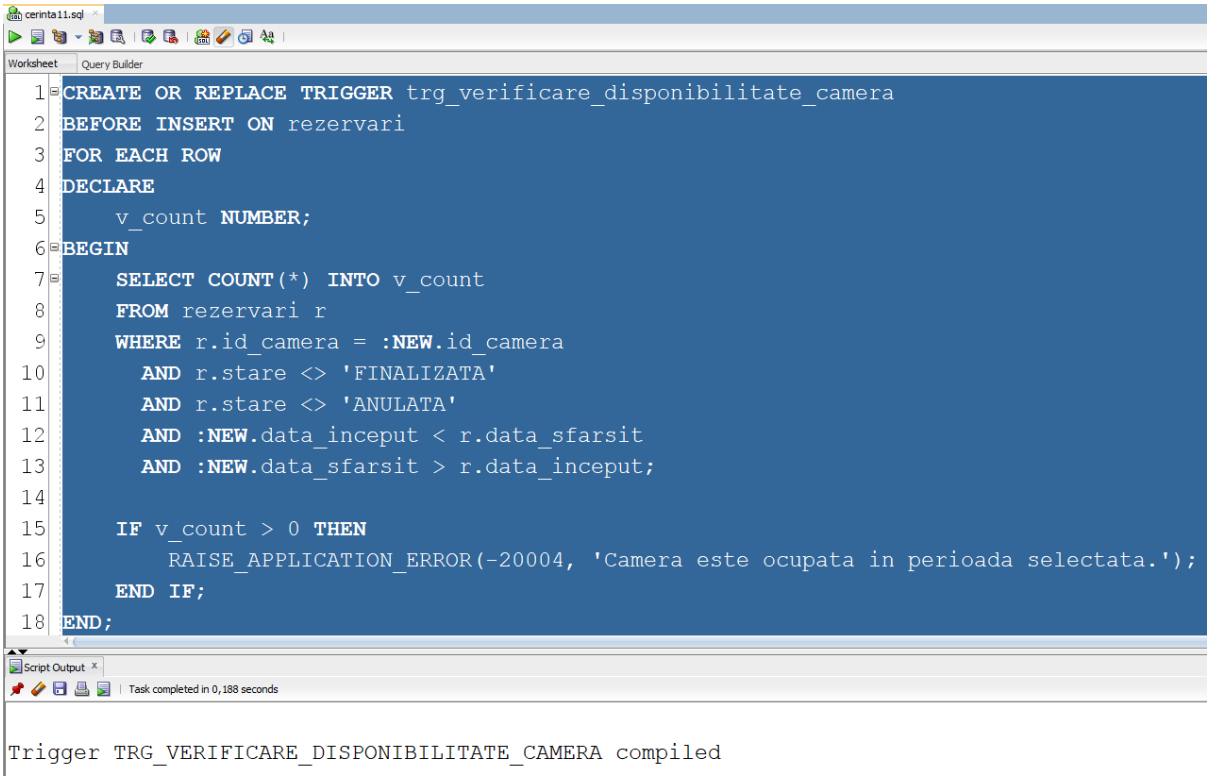
Cerința 11

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

Să se creeze un trigger care **nu permite inserarea unei rezervări dacă este deja ocupată camera** în acea perioadă de o rezervare care nu este finalizată sau anulată.

```
CREATE OR REPLACE TRIGGER
trg_verificare_disponibilitate_camera
BEFORE INSERT ON rezervari
FOR EACH ROW
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count
    FROM rezervari r
    WHERE r.id_camera = :NEW.id_camera
      AND r.stare <> 'FINALIZATA'
      AND r.stare <> 'ANULATA'
      AND :NEW.data_inceput < r.data_sfarsit
      AND :NEW.data_sfarsit > r.data_inceput;

    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20004, 'Camera este ocupata
in perioada selectata.');
    END IF;
```

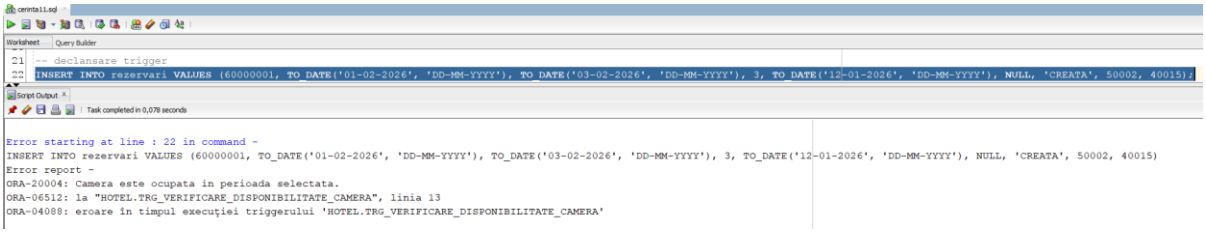
```
END;
/
```



```
CREATE OR REPLACE TRIGGER trg_verificare_disponibilitate_camera
BEFORE INSERT ON rezervari
FOR EACH ROW
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count
    FROM rezervari r
    WHERE r.id_camera = :NEW.id_camera
      AND r.stare <> 'FINALIZATA'
      AND r.stare <> 'ANULATA'
      AND :NEW.data_inceput < r.data_sfarsit
      AND :NEW.data_sfarsit > r.data_inceput;

    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20004, 'Camera este ocupata in perioada selectata.');
    END IF;
END;
```

```
Trigger TRG_VERIFICARE_DISPONIBILITATE_CAMERA compiled
```

```
-- declansare trigger

INSERT INTO rezervari VALUES (60000001, TO_DATE('01-02-2026',
'DD-MM-YYYY'), TO_DATE('03-02-2026', 'DD-MM-YYYY'), 3,
TO_DATE('12-01-2026', 'DD-MM-YYYY'), NULL, 'CREATA', 50002,
40015);
```



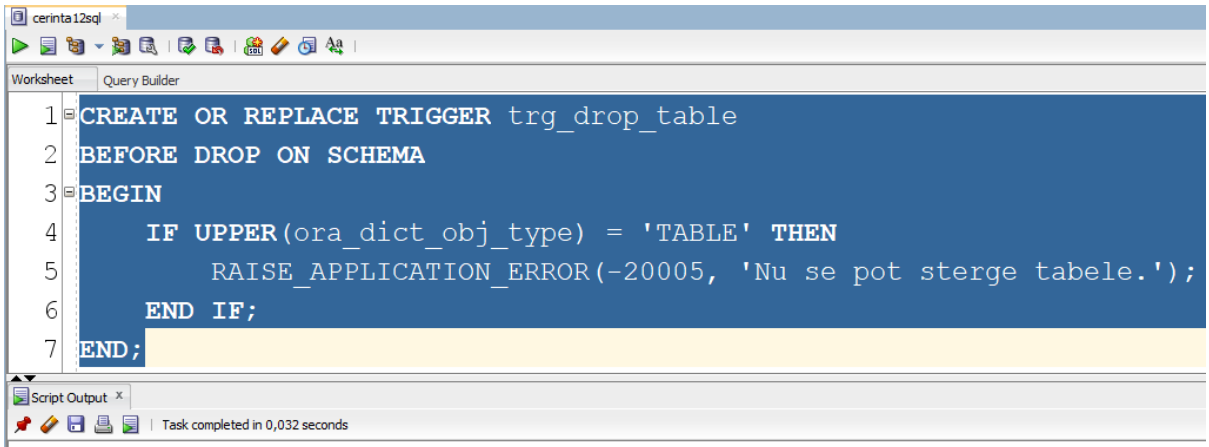
```
-- declansare trigger
INSERT INTO rezervari VALUES (60000001, TO_DATE('01-02-2026', 'DD-MM-YYYY'), TO_DATE('03-02-2026', 'DD-MM-YYYY'), 3, TO_DATE('12-01-2026', 'DD-MM-YYYY'), NULL, 'CREATA', 50002, 40015);

Error starting at line : 22 in command -
INSERT INTO rezervari VALUES (60000001, TO_DATE('01-02-2026', 'DD-MM-YYYY'), TO_DATE('03-02-2026', 'DD-MM-YYYY'), 3, TO_DATE('12-01-2026', 'DD-MM-YYYY'), NULL, 'CREATA', 50002, 40015)
Error report -
ORA-20004: Camera este ocupata in perioada selectata.
ORA-06512: la "HOTEL.TRG_VERIFICARE_DISPONIBILITATE_CAMERA", linia 13
ORA-04088: eroare în timpul execuţiei triggerului 'HOTEL.TRG_VERIFICARE_DISPONIBILITATE_CAMERA'
```

Cerința 12

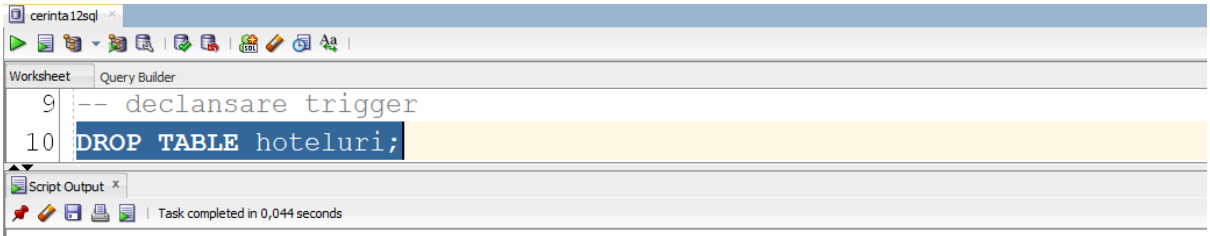
Definiți un trigger de tip LDD. Declanșați trigger-ul.

Să se creeze un trigger care **nu permite ștergerea tabelelor**.

```
CREATE OR REPLACE TRIGGER trg_drop_table
BEFORE DROP ON SCHEMA
BEGIN
    IF UPPER(ora_dict_obj_type) = 'TABLE' THEN
        RAISE_APPLICATION_ERROR(-20005, 'Nu se pot sterge
tabele.');
    END IF;
END;
/
```




```
-- declansare trigger
DROP TABLE hoteluri;
```




```
Error starting at line : 10 in command -
DROP TABLE hoteluri
Error report -
ORA-04088: eroare în timpul execuţiei triggerului 'HOTEL.TRG_DROP_TABLE'
ORA-00604: eroare apărută la SQL recursiv nivelul 1
ORA-20005: Nu se pot sterge tabele.
ORA-06512: la linia 3
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.
```

Cerința 13

Formulați în limbaj natural o problemă pe care să o rezolvați folosind un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).

Să se creeze un pachet pentru **gestionarea rezervărilor**, care să includă următoarele:

Tipuri de date:

- un tip de date **RECORD** care stochează toate informațiile aferente unei rezervări
- un tip de date colecție **VARRAY** care conține procentele reducerilor aplicate unei rezervări

Funcții:

- o funcție care verifică dacă o rezervare **este validă** (primind ca parametru rezervarea sub forma unui RECORD) din punct de vedere al regulilor de validare și integritate impuse de baza de date
- o funcție care calculează **numărul de nopți** dintr-o rezervare validă
- o funcție care calculează **procentul total al reducerilor** aplicate (având ca parametru un VARRAY cu valori care trebuie însumate)
- o funcție care determină dacă o rezervare validă **este activă** (dacă nu este anulată și data curentă se află în perioada sejurului)

Proceduri:

- o procedură care calculează și salvează (în parametrul RECORD transmis ca referință) **totalul de plată** pentru o rezervare validă (pe baza numărului de nopți și tarifului camerei pe noapte, fără serviciile suplimentare)
- o procedură care primește o rezervare validă și colecția de reduceri și **aplică toate reducerile** la totalul de plată al rezervării
- o procedură care **permite schimbarea camerei** cu o cameră din același hotel pentru o rezervare validă și activă, ce primește ca parametri rezervarea, reducerile și numărul camerei noi (dacă acea cameră este liberă și poate caza numărul dat de persoane, se recalculează totalul de plată aplicând noul tarif de noapte pentru restul sejurului, începând cu noaptea următoare)

- o procedură care **încarcă o rezervare din baza de date într-un RECORD**
- o procedură care **inserează o rezervare** validă primită ca parametru **în baza de date sau îi actualizează datele** dacă ea există deja
- o procedură care **afișează pe ecran toate informațiile unei rezervări** din baza de date

```
CREATE OR REPLACE PACKAGE pkg_rezervari
IS
    TYPE rezervare_rec IS RECORD (
        id_rezervare rezervari.id_rezervare%TYPE,
        data_inceput rezervari.data_inceput%TYPE,
        data_sfarsit rezervari.data_sfarsit%TYPE,
        numar_persoane rezervari.numar_persoane%TYPE,
        data_rezervare rezervari.data_rezervare%TYPE,
        total_plata rezervari.total_plata%TYPE,
        stare rezervari.stare%TYPE,
        id_client rezervari.id_client%TYPE,
        id_camera rezervari.id_camera%TYPE
    );

    TYPE reduceri_varray IS VARRAY(10) OF NUMBER;

    c_err_rezervare_invalida CONSTANT NUMBER := -20006;
    c_msg_rezervare_invalida CONSTANT VARCHAR2(30) :=
'Rezervarea este invalida.';

    FUNCTION este_valida (
```

```
    p_rezervare IN rezervare_rec
) RETURN BOOLEAN;

FUNCTION calculeaza_numar_nopti (
    p_rezervare IN rezervare_rec
) RETURN NUMBER;

FUNCTION calculeaza_reducere_totala (
    p_reduceri IN reduceri_varray
) RETURN NUMBER;

FUNCTION este_activa (
    p_rezervare IN rezervare_rec
) RETURN BOOLEAN;

PROCEDURE calculeaza_total_plata (
    p_rezervare IN OUT rezervare_rec
);

PROCEDURE aplica_reduceri (
    p_rezervare IN OUT rezervare_rec,
    p_reduceri IN reduceri_varray
);

PROCEDURE schimba_camera (
```

```
        p_rezervare IN OUT rezervare_rec,
        p_reduceri IN reduceri_varray,
        p_numar_camera IN camere.numar_camera%TYPE
    );

    PROCEDURE select_rezervare (
        p_id_rezervare IN rezervari.id_rezervare%TYPE,
        p_rezervare OUT rezervare_rec
    );

    PROCEDURE insert_sau_update_rezervare (
        p_rezervare IN rezervare_rec
    );

    PROCEDURE afisare_rezervare (
        p_id_rezervare rezervari.id_rezervare%TYPE
    );
END pkg_rezervari;
/

CREATE OR REPLACE PACKAGE BODY pkg_rezervari
IS
    FUNCTION este_valida (
        p_rezervare IN rezervare_rec
    ) RETURN BOOLEAN
```

```
    IS
        v_count NUMBER;
    BEGIN
        IF p_rezervare.id_rezervare IS NULL
           OR p_rezervare.data_inceput IS NULL
           OR p_rezervare.data_sfarsit IS NULL
           OR p_rezervare.numar_persoane IS NULL
           OR p_rezervare.data_rezervare IS NULL
           OR p_rezervare.stare IS NULL
           OR p_rezervare.id_client IS NULL
           OR p_rezervare.id_camera IS NULL THEN
            RETURN FALSE;
        END IF;

        IF p_rezervare.data_inceput >=
p_rezervare.data_sfarsit THEN
            RETURN FALSE;
        END IF;

        IF p_rezervare.numar_persoane <= 0 THEN
            RETURN FALSE;
        END IF;

        IF p_rezervare.data_rezervare >
p_rezervare.data_inceput THEN
```

```
            RETURN FALSE;
        END IF;

        IF UPPER(p_rezervare.stare) NOT IN ('CREATA',
'CONFIRMATA', 'ANULATA', 'FINALIZATA') THEN
            RETURN FALSE;
        END IF;

        SELECT COUNT(*) INTO v_count
        FROM clienti
        WHERE id_client = p_rezervare.id_client;

        IF v_count = 0 THEN
            RETURN FALSE;
        END IF;

        SELECT COUNT(*) INTO v_count
        FROM camere
        WHERE id_camera = p_rezervare.id_camera;

        IF v_count = 0 THEN
            RETURN FALSE;
        END IF;

        RETURN TRUE;
```

```
    END este_valida;

    FUNCTION calculeaza_numar_nopti (
        p_rezervare IN rezervare_rec
    ) RETURN NUMBER
    IS
    BEGIN
        IF NOT este_valida(p_rezervare) THEN
            RAISE_APPLICATION_ERROR(c_err_rezervare_invalida,
c_msg_rezervare_invalida);
        END IF;

        RETURN TRUNC(p_rezervare.data_sfarsit) -
TRUNC(p_rezervare.data_inceput);
    END calculeaza_numar_nopti;

    FUNCTION calculeaza_reducere_totala (
        p_reduceri IN reduceri_varray
    ) RETURN NUMBER
    IS
        v_total NUMBER;
    BEGIN
        IF p_reduceri IS NULL THEN
            RETURN 0;
        END IF;
```

```
        v_total := 0;
        FOR i IN 1..p_reduceri.COUNT LOOP
            IF p_reduceri(i) < 0 OR p_reduceri(i) > 100 THEN
                RAISE_APPLICATION_ERROR(-20007, 'Procent de
reducere invalid: valoarile trebuie sa fie intre 0 si 100.');
            END IF;

            v_total := v_total + p_reduceri(i);
        END LOOP;

        IF v_total > 100 THEN
            v_total := 100;
        END IF;

        RETURN v_total;
    END calculeaza_reducere_totala;

    FUNCTION este_activa (
        p_rezervare IN rezervare_rec
    ) RETURN BOOLEAN
    IS
    BEGIN
        IF NOT este_valida(p_rezervare) THEN
```

```
            RAISE_APPLICATION_ERROR(c_err_rezervare_invalida,
c_msg_rezervare_invalida);
        END IF;

        IF TRUNC(SYSDATE) >= p_rezervare.data_inceput AND
TRUNC(SYSDATE) <= p_rezervare.data_sfarsit AND
UPPER(p_rezervare.stare) <> 'ANULATA' THEN
            RETURN TRUE;
        END IF;

        RETURN FALSE;
    END este_activa;

    PROCEDURE calculeaza_total_plata (
        p_rezervare IN OUT rezervare_rec
    )
    IS
        v_pret_noapte camere.pret_noapte%TYPE;
    BEGIN
        IF NOT este_valida(p_rezervare) THEN
            RAISE_APPLICATION_ERROR(c_err_rezervare_invalida,
c_msg_rezervare_invalida);
        END IF;

        SELECT pret_noapte INTO v_pret_noapte
        FROM camere
```

```
        WHERE id_camera = p_rezervare.id_camera;

        p_rezervare.total_plata :=
calculeaza_numar_nopti(p_rezervare) * v_pret_noapte;
    END calculeaza_total_plata;

    PROCEDURE aplica_reduceri (
        p_rezervare IN OUT rezervare_rec,
        p_reduceri IN reduceri_varray
    )
    IS
    BEGIN
        IF NOT este_valida(p_rezervare) THEN
            RAISE_APPLICATION_ERROR(c_err_rezervare_invalida,
c_msg_rezervare_invalida);
        END IF;

        IF p_rezervare.total_plata IS NULL THEN
            RAISE_APPLICATION_ERROR(-20008, 'Reducerile nu se
pot aplica inaintea calculului pretului de baza.');
        END IF;

        p_rezervare.total_plata := p_rezervare.total_plata *
(1 - calculeaza_reducere_totala(p_reduceri) / 100);
    END aplica_reduceri;
```

```
    PROCEDURE schimba_camera (
        p_rezervare IN OUT rezervare_rec,
        p_reduceri IN reduceri_varray,
        p_numar_camera IN camere.numar_camera%TYPE
    )
    IS
        v_id_hotel hoteluri.id_hotel%TYPE;
        v_id_camera camere.id_camera%TYPE;
        v_nopti_inainte NUMBER;
        v_nopti_dupa NUMBER;
        v_pret_inainte camere.pret_noapte%TYPE;
        v_pret_dupa camere.pret_noapte%TYPE;
        v_numar_maxim_persoane
camere.numar_maxim_persoane%TYPE;
        v_count NUMBER;
        v_data_noua DATE;
    BEGIN
        IF NOT este_activa(p_rezervare) THEN
            RAISE_APPLICATION_ERROR(-20009, 'Rezervarea nu
este activa.');
        END IF;

        v_data_noua := TRUNC(SYSDATE) + 1;
```

```
        SELECT id_hotel, pret_noapte INTO v_id_hotel,
v_pret_inainte
        FROM camere
        WHERE id_camera = p_rezervare.id_camera;

        SELECT id_camera, pret_noapte, numar_maxim_persoane
INTO v_id_camera, v_pret_dupa, v_numar_maxim_persoane
        FROM camere
        WHERE id_hotel = v_id_hotel AND numar_camera =
p_numar_camera;

        IF p_rezervare.numar_persoane > v_numar_maxim_persoane
THEN
            RAISE_APPLICATION_ERROR(-20010, 'Numarul de
persoane este mai mare decat capacitatea camerei.');
        END IF;

        SELECT COUNT(*) INTO v_count
        FROM rezervari r
        WHERE r.id_camera = v_id_camera
          AND r.stare <> 'ANULATA'
          AND r.id_rezervare <> p_rezervare.id_rezervare
          AND r.data_inceput < p_rezervare.data_sfarsit
          AND r.data_sfarsit > v_data_noua;

        IF v_count > 0 THEN
```

```
            RAISE_APPLICATION_ERROR(-20011, 'Camera selectata
nu este libera in perioada ramasa.');
        END IF;

        v_nopti_inainte := v_data_noua -
p_rezervare.data_inceput;
        v_nopti_dupa := p_rezervare.data_sfarsit -
v_data_noua;

        p_rezervare.id_camera := v_id_camera;

        p_rezervare.total_plata := v_nopti_inainte *
v_pret_inainte + v_nopti_dupa * v_pret_dupa;

        aplica_reduceri(p_rezervare, p_reduceri);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20012, 'Camera nu
exista.');
    END schimba_camera;

    PROCEDURE select_rezervare (
        p_id_rezervare IN rezervari.id_rezervare%TYPE,
        p_rezervare OUT rezervare_rec
    )
    IS
```

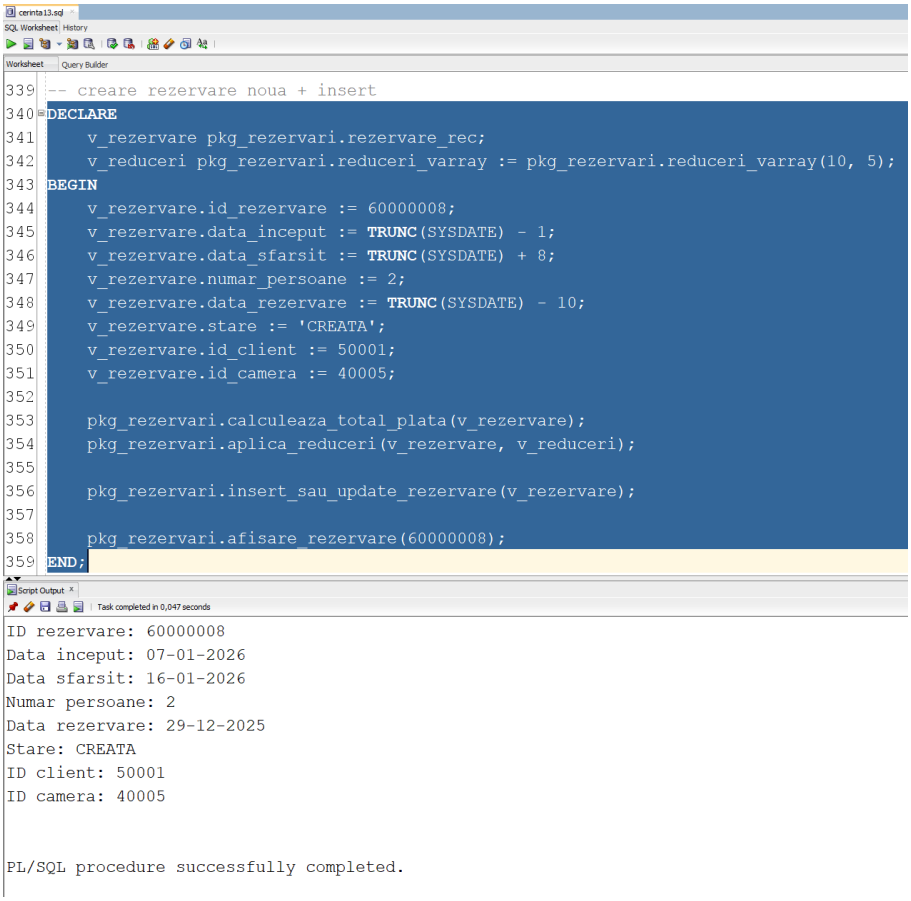
```
    BEGIN
        IF p_id_rezervare IS NULL THEN
            RAISE_APPLICATION_ERROR(-20012, 'ID rezervare
invalid.');
        END IF;

        SELECT id_rezervare, data_inceput, data_sfarsit,
numar_persoane, data_rezervare, total_plata, stare, id_client,
id_camera
        INTO p_rezervare.id_rezervare,
p_rezervare.data_inceput, p_rezervare.data_sfarsit,
p_rezervare.numar_persoane, p_rezervare.data_rezervare,
             p_rezervare.total_plata, p_rezervare.stare,
p_rezervare.id_client, p_rezervare.id_camera
        FROM rezervari
        WHERE id_rezervare = p_id_rezervare;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20013, 'Rezervarea nu
exista.');
    END select_rezervare;

    PROCEDURE insert_sau_update_rezervare (
        p_rezervare IN rezervare_rec
    )
    IS
        v_count NUMBER;
```

```
    BEGIN
        IF NOT este_valida(p_rezervare) THEN
            RAISE_APPLICATION_ERROR(c_err_rezervare_invalida,
c_msg_rezervare_invalida);
        END IF;

        SELECT COUNT(*) INTO v_count
        FROM rezervari
        WHERE id_rezervare = p_rezervare.id_rezervare;

        IF v_count = 0 THEN
            INSERT INTO rezervari(id_rezervare, data_inceput,
data_sfarsit, numar_persoane, data_rezervare, total_plata,
stare, id_client, id_camera)
            VALUES(p_rezervare.id_rezervare,
p_rezervare.data_inceput, p_rezervare.data_sfarsit,
p_rezervare.numar_persoane,
                   p_rezervare.data_rezervare,
p_rezervare.total_plata, p_rezervare.stare,
p_rezervare.id_client, p_rezervare.id_camera);
        ELSE
            UPDATE rezervari
            SET data_inceput = p_rezervare.data_inceput,
                data_sfarsit = p_rezervare.data_sfarsit,
                numar_persoane = p_rezervare.numar_persoane,
                data_rezervare = p_rezervare.data_rezervare,
                total_plata = p_rezervare.total_plata,
```

```
                stare = p_rezervare.stare,
                id_client = p_rezervare.id_client,
                id_camera = p_rezervare.id_camera
            WHERE id_rezervare = p_rezervare.id_rezervare;
        END IF;
    END insert_sau_update_rezervare;

    PROCEDURE afisare_rezervare (
        p_id_rezervare rezervari.id_rezervare%TYPE
    )
    IS
        v_rezervare rezervare_rec;
    BEGIN
        select_rezervare(p_id_rezervare, v_rezervare);

        DBMS_OUTPUT.PUT_LINE('ID rezervare: ' ||
v_rezervare.id_rezervare);
        DBMS_OUTPUT.PUT_LINE('Data inceput: ' ||
TO_CHAR(v_rezervare.data_inceput, 'DD-MM-YYYY'));
        DBMS_OUTPUT.PUT_LINE('Data sfarsit: ' ||
TO_CHAR(v_rezervare.data_sfarsit, 'DD-MM-YYYY'));
        DBMS_OUTPUT.PUT_LINE('Numar persoane: ' ||
v_rezervare.numar_persoane);
        DBMS_OUTPUT.PUT_LINE('Data rezervare: ' ||
TO_CHAR(v_rezervare.data_rezervare, 'DD-MM-YYYY'));
        DBMS_OUTPUT.PUT_LINE('Stare: ' || v_rezervare.stare);
```

```
        DBMS_OUTPUT.PUT_LINE('ID client: ' ||
v_rezervare.id_client);
        DBMS_OUTPUT.PUT_LINE('ID camera: ' ||
v_rezervare.id_camera);
    END;
END pkg_rezervari;
/


-- creare rezervare noua + insert
DECLARE
    v_rezervare pkg_rezervari.rezervare_rec;
    v_reduceri pkg_rezervari.reduceri_varray :=
pkg_rezervari.reduceri_varray(10, 5);
BEGIN
    v_rezervare.id_rezervare := 60000008;
    v_rezervare.data_inceput := TRUNC(SYSDATE) - 1;
    v_rezervare.data_sfarsit := TRUNC(SYSDATE) + 8;
    v_rezervare.numar_persoane := 2;
    v_rezervare.data_rezervare := TRUNC(SYSDATE) - 10;
    v_rezervare.stare := 'CREATA';
    v_rezervare.id_client := 50001;
    v_rezervare.id_camera := 40005;


    pkg_rezervari.calculeaza_total_plata(v_rezervare);
    pkg_rezervari.aplica_reduceri(v_rezervare, v_reduceri);
```

```
    pkg_rezervari.insert_sau_update_rezervare(v_rezervare);

    pkg_rezervari.afisare_rezervare(60000008);
END;
/
```

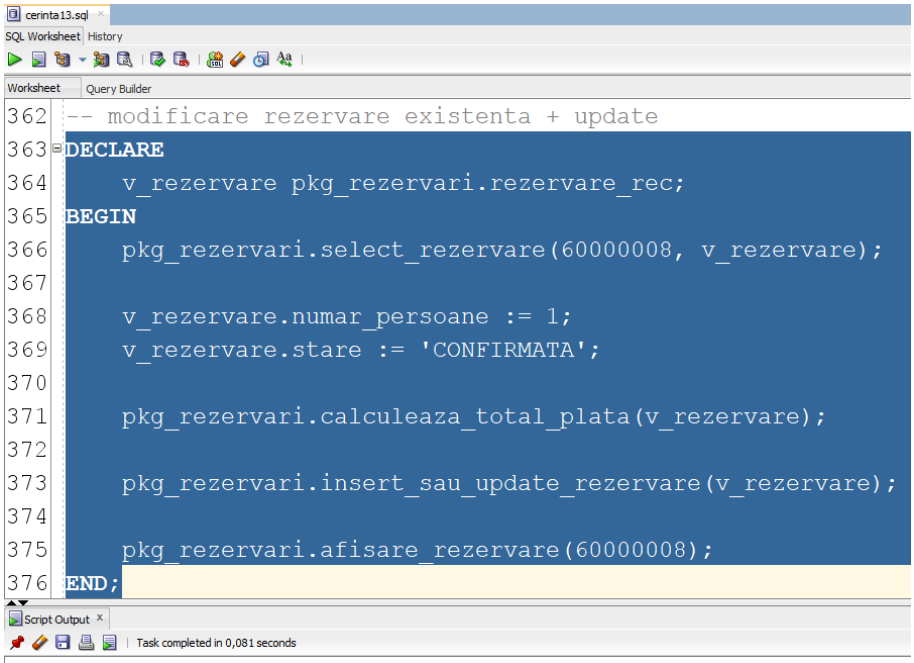


```
-- creare rezervare noua + insert
DECLARE
    v_rezervare pkg_rezervari.rezervare_rec;
    v_reduceri pkg_rezervari.reduceri_varray := pkg_rezervari.reduceri_varray(10, 5);
BEGIN
    v_rezervare.id_rezervare := 60000008;
    v_rezervare.data_inceput := TRUNC(SYSDATE) - 1;
    v_rezervare.data_sfarsit := TRUNC(SYSDATE) + 8;
    v_rezervare.numar_persoane := 2;
    v_rezervare.data_rezervare := TRUNC(SYSDATE) - 10;
    v_rezervare.stare := 'CREATA';
    v_rezervare.id_client := 50001;
    v_rezervare.id_camera := 40005;

    pkg_rezervari.calculeaza_total_plata(v_rezervare);
    pkg_rezervari.aplica_reduceri(v_rezervare, v_reduceri);

    pkg_rezervari.insert_sau_update_rezervare(v_rezervare);

    pkg_rezervari.afisare_rezervare(60000008);
END;
```

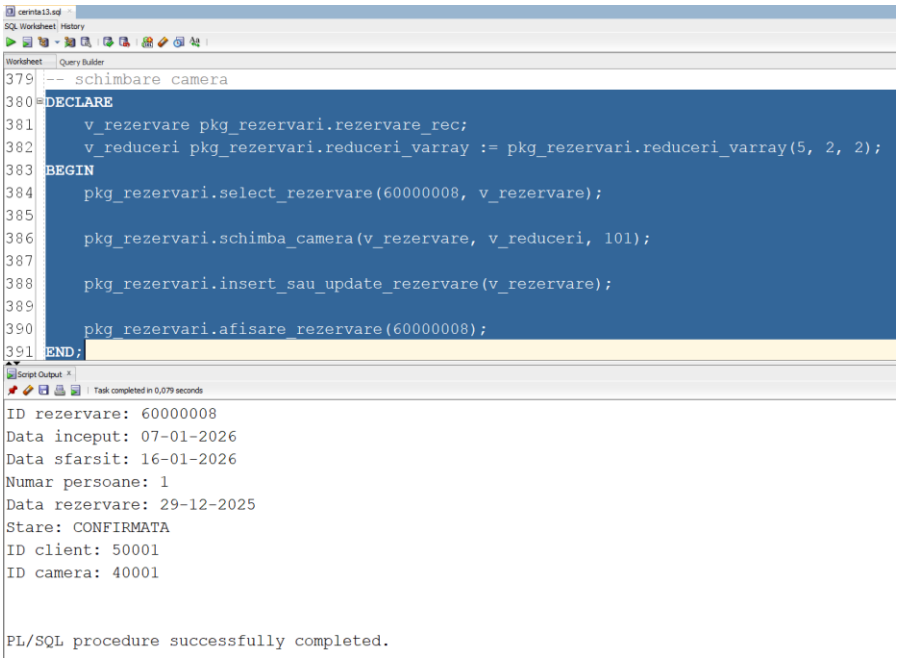
```
ID rezervare: 60000008
Data inceput: 07-01-2026
Data sfarsit: 16-01-2026
Numar persoane: 2
Data rezervare: 29-12-2025
Stare: CREATA
ID client: 50001
ID camera: 40005


PL/SQL procedure successfully completed.
```

```
-- modificare rezervare existenta + update
DECLARE
    v_rezervare pkg_rezervari.rezervare_rec;
BEGIN
    pkg_rezervari.select_rezervare(60000008, v_rezervare);
```

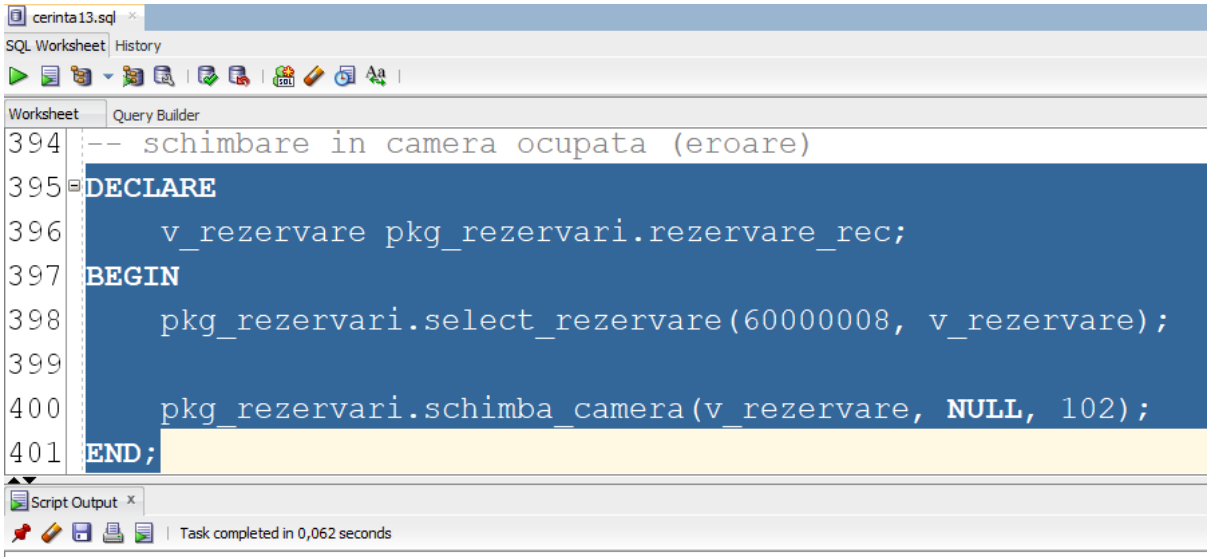
```
    v_rezervare.numar_persoane := 1;
    v_rezervare.stare := 'CONFIRMATA';

    pkg_rezervari.calculeaza_total_plata(v_rezervare);

    pkg_rezervari.insert_sau_update_rezervare(v_rezervare);

    pkg_rezervari.afisare_rezervare(60000008);
END;
/
```



```
-- schimbare camera
DECLARE
    v_rezervare pkg_rezervari.rezervare_rec;
    v_reduceri pkg_rezervari.reduceri_varray :=
pkg_rezervari.reduceri_varray(5, 2, 2);
BEGIN
    pkg_rezervari.select_rezervare(60000008, v_rezervare);

    pkg_rezervari.schimba_camera(v_rezervare, v_reduceri,
101);

    pkg_rezervari.insert_sau_update_rezervare(v_rezervare);

    pkg_rezervari.afisare_rezervare(60000008);
END;
/
```

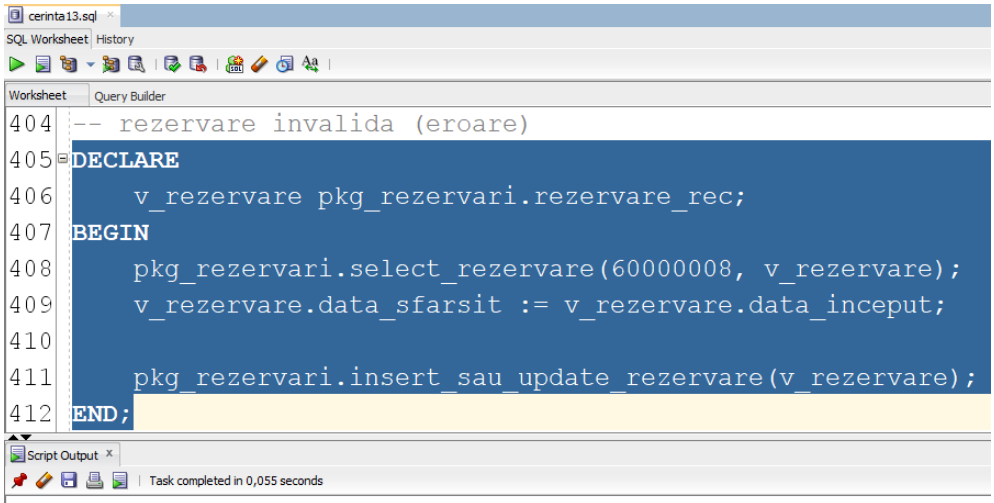


```
-- schimbare in camera ocupata (eroare)
DECLARE
    v_rezervare pkg_rezervari.rezervare_rec;
BEGIN
    pkg_rezervari.select_rezervare(60000008, v_rezervare);

    pkg_rezervari.schimba_camera(v_rezervare, NULL, 102);
END;
/
```




```
Error starting at line : 395 in command -
DECLARE
    v_rezervare pkg_rezervari.rezervare_rec;
BEGIN
    pkg_rezervari.select_rezervare(60000008, v_rezervare);

    pkg_rezervari.schimba_camera(v_rezervare, NULL, 102);
END;
Error report -
ORA-20011: Camera selectata nu este libera in perioada ramasa.
ORA-06512: la "HOTEL.PKG_REZERVARI", linia 186
ORA-06512: la linia 6
```

```
-- rezervare invalida (eroare)
DECLARE
    v_rezervare pkg_rezervari.rezervare_rec;
BEGIN
    pkg_rezervari.select_rezervare(60000008, v_rezervare);
    v_rezervare.data_sfarsit := v_rezervare.data_inceput;


    pkg_rezervari.insert_sau_update_rezervare(v_rezervare);
END;
/
```




```
Error starting at line : 405 in command -
DECLARE
    v_rezervare pkg_rezervari.rezervare_rec;
BEGIN
    pkg_rezervari.select_rezervare(60000008, v_rezervare);
    v_rezervare.data_sfarsit := v_rezervare.data_inceput;

    pkg_rezervari.insert_sau_update_rezervare(v_rezervare);
END;
Error report -
ORA-20006: Rezervarea este invalida.
ORA-06512: la "HOTEL.PKG_REZERVARI", linia 229
ORA-06512: la linia 7
```

```
COMMIT;
```