

PROIECT DATAWAREHOUSE
SISTEM DE GESTIUNE AL REZERVĂRILOR HOTELIERE

MODUL ANALIZĂ

1. Descrierea modelului ales și a obiectivelor aplicației.....	3
2. Diagramele bazei de date OLTP	
a. Diagrama entitate – relație a bazei de date OLTP.....	4
b. Diagrama conceptuală a bazei de date OLTP.....	5
3. Diagrama stea/fulg a bazei de date depozit.....	7
4. Descrierea câmpurilor necesare pentru fiecare tabel din baza de date depozit și modul de populare al acestora cu informații din baza de date OLTP	8
5. Identificarea constrângerilor specifice depozitelor de date ce trebuie definite.....	11
6. Identificarea indecșilor specifici depozitelor de date ce trebuie definiți asupra modelului. Formularea unei cereri în limbaj natural care va determina utilizarea indecșilor specificați și va fi implementată în următoarea etapă.....	12
7. Identificarea obiectelor de tip dimensiune ce trebuie definite asupra modelului.....	13
8. Identificarea tabelelor care vor fi partiționate și a tipului de partiționare. Formularea unei cereri în limbaj natural care va determina utilizarea lor și va fi implementată în următoarea etapă.....	14
9. Formularea în limbaj natural a unei cereri SQL complexe care va fi optimizată în următoarea etapă, folosind tehnici specifice bazelor de date depozit.Precizarea tehnicilor de optimizare ce ar putea fi utilizate pentru această cerere particular.....	15
10. Formularea în limbaj natural a cel puțin 5 cereri cu grad de complexitate diferit, concretizate în rapoarte (grafice) ce vor fi create în următoarele etape.....	15

MODUL IMPLEMENTARE BAZA DE DATE

1. Crearea bazei de date OLTP și a utilizatorilor.	16
2. Generarea datelor și inserarea acestora în tabele.....	23
3. Crearea bazei de date depozit și a utilizatorilor.....	31
4. Popularea cu informații a bazei de date depozit folosind ca sursă datele din baza de date OLTP.....	38
5. Definirea constrângerilor.....	47
6. Definirea indecșilor și a cererilor SQL însoțite de planul de execuție al acestora.....	48
7. Definirea obiectelor de tip dimensiune, validarea acestora.....	50
8. Definirea partițiilor; definirea cererilor SQL însoțite de planul de execuție al acestora din care să reiasă ca optimizorul utilizează eficient partițiile.....	52
9. Optimizarea cererii SQL propusă în etapa de analiză	
a. planul de execuție ales de optimizorul bazat pe cost (explicație etape parcurse).....	52
b. sugestii de optimizare a cererii, specificând planul de execuție obținut.....	53
10. Crearea rapoartelor cu complexitate diferită (la acest nivel vor fi scripturi SQL, fără reprezentare grafică).....	55

ETAPA IMPLEMENTARE APLICATIE

1. Modul aplicație prin care se introduc și gestionează informații la nivelul bazei de date OLTP.....	56
2. Posibilitatea de vizualizare a datelor introduse/actualizate la nivelul bazei de date OLTP și posibilitatea de a verifica propagarea acestor operații asupra datelor din baza de date depozit.....	66
3. Rapoartele grafice asociate cererilor definite în etapele anterioare (punctul 10).....	70

PROIECT DATA WAREHOUSE

~ Modul analiza ~

1. Descrierea modelului ales și a obiectivelor aplicației.

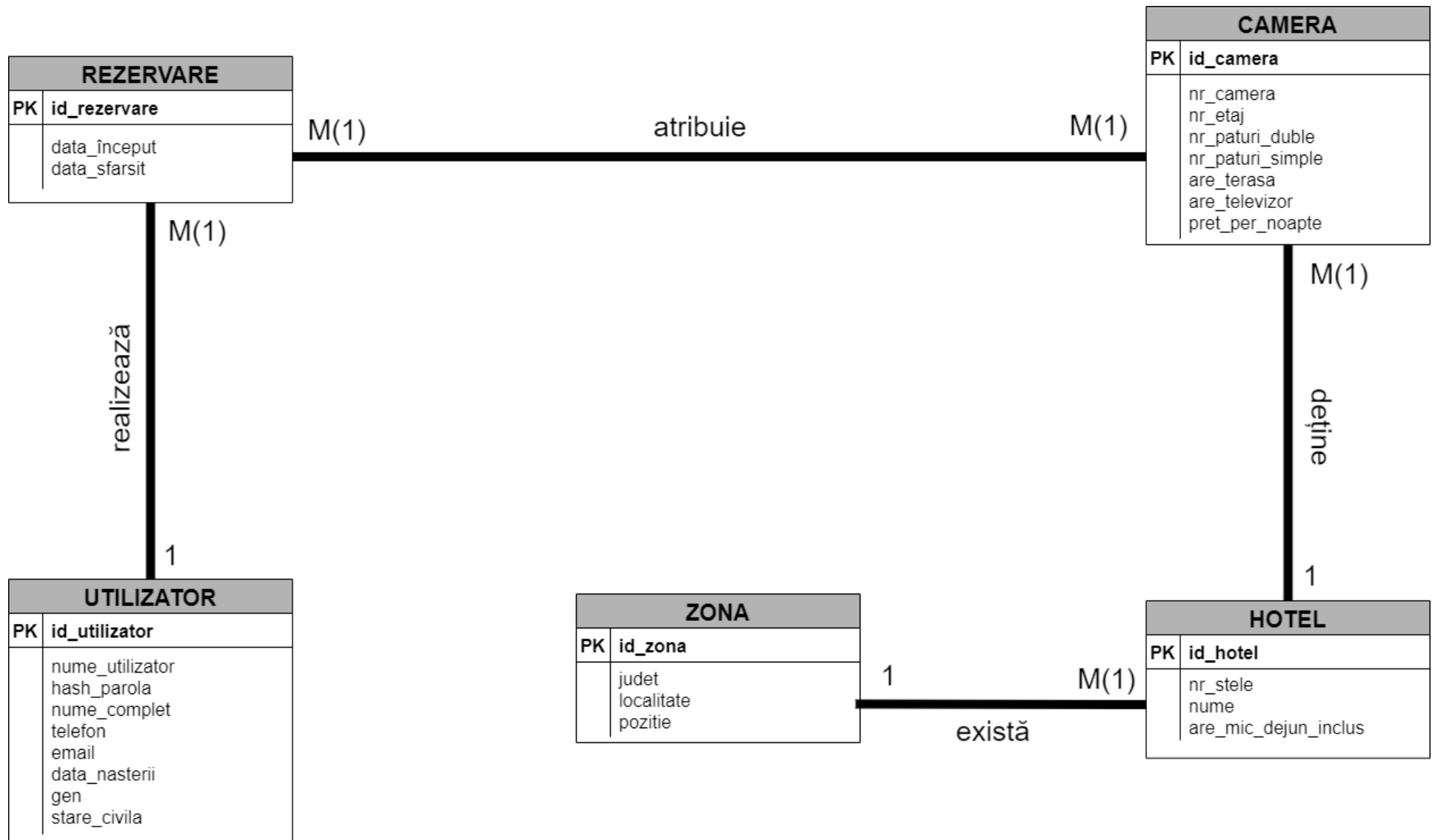
Pentru a înțelege funcționalitatea aplicației noastre, vom începe cu ***descrierea modelului***. Aplicația oferă posibilitatea utilizatorilor de a intra în aplicație prin intermediul completării unor detalii cu caracter personal cum ar fi numărul de telefon, adresa de email, data_nasterii, genul, starea_civilă, etc.

Clientul poate realiza una sau mai multe rezervări. Aceste rezervări atribuie una sau mai multe camere în funcție de perioada disponibilă. Un hotel partener al aplicației noastre deține un număr de stele și poate avea mai multe camere. O camera poate avea o serie de caracteristici importante cum ar fi etajul, numărul de paturi duble sau simple, terasă, televizor. La finalul sejurului, clientul trebuie să plătească o sumă de bani în funcție de prețul camerei pe noapte și de numărul de zile de cazare. Fiecare hotel se diferențiază în funcție de zona în care se află.

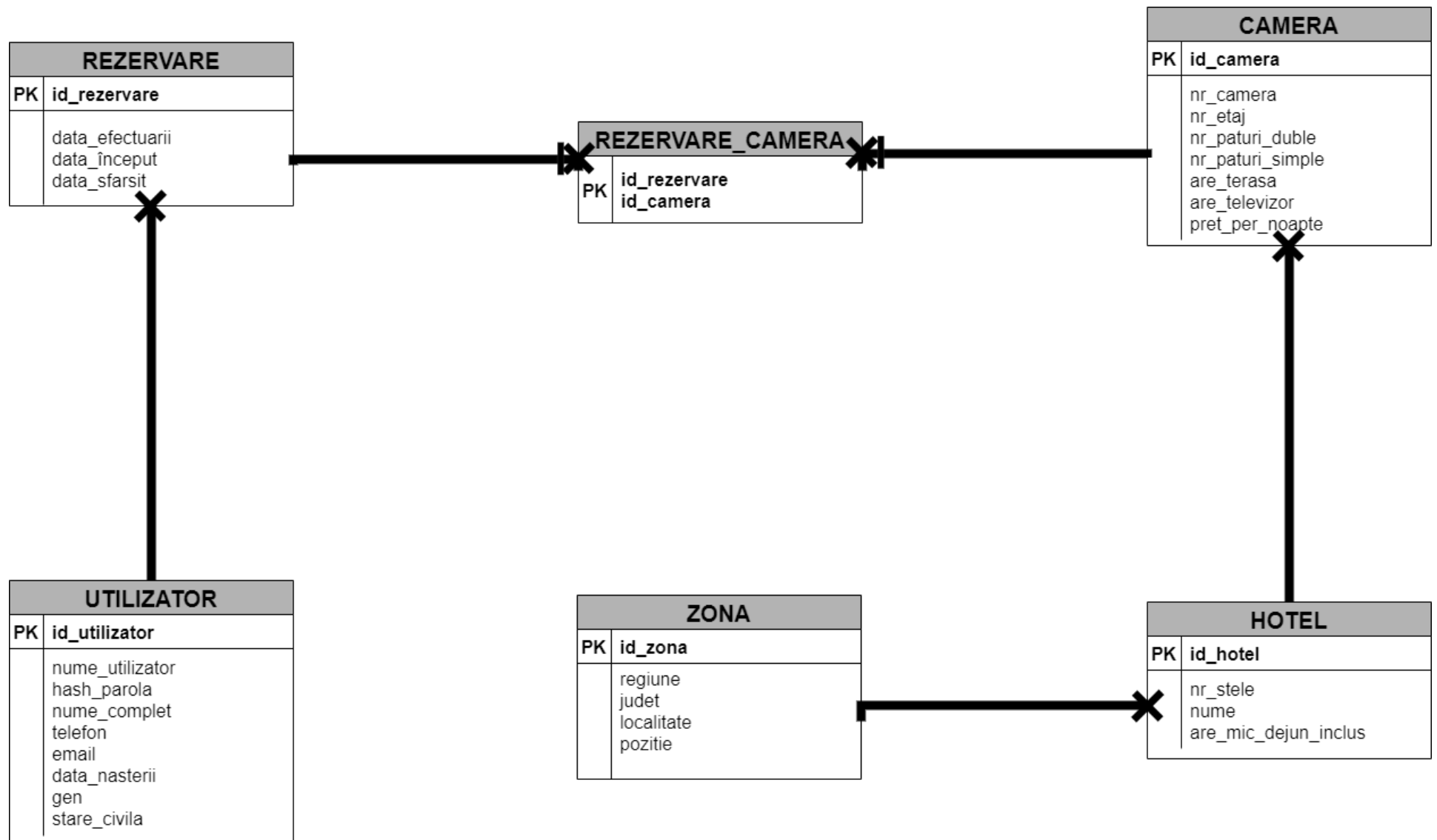
Obiectivele aplicației sunt de a oferi o interfață cât mai intuitivă clientului pentru a putea să își aleagă cu ușurință locul unde dorește să își petreacă vacanța și de a oferi managerilor o serie de rapoarte bogate în informații utile pentru a putea să își înțeleagă mai bine afacerea. Printre aceste rapoarte putem întâlni informații cu privire la perioadele de aglomerare a clienților, vârsta clienților majoritari în funcție de perioadă și informații cu privire la zonele de amplasare a viitoarelor hoteluri, etc.

2. Diagramele bazei de date OLTP

a. Diagrama entitate – relație a bazei de date OLTP.



b. Diagrama conceptuală a bazei de date OLTP.



Schemele relaționale corespunzătoare acestei diagrame conceptuale sunt următoarele:

UTILIZATOR (#id_utilizator, nume_utilizator, hash_parola, nume_complet, telefon, email, data_nasterii, gen, stare_civilă)

REZERVARE (#id_rezervare, id_utilizator, data_început, data_sfârșit, data_efectuării)

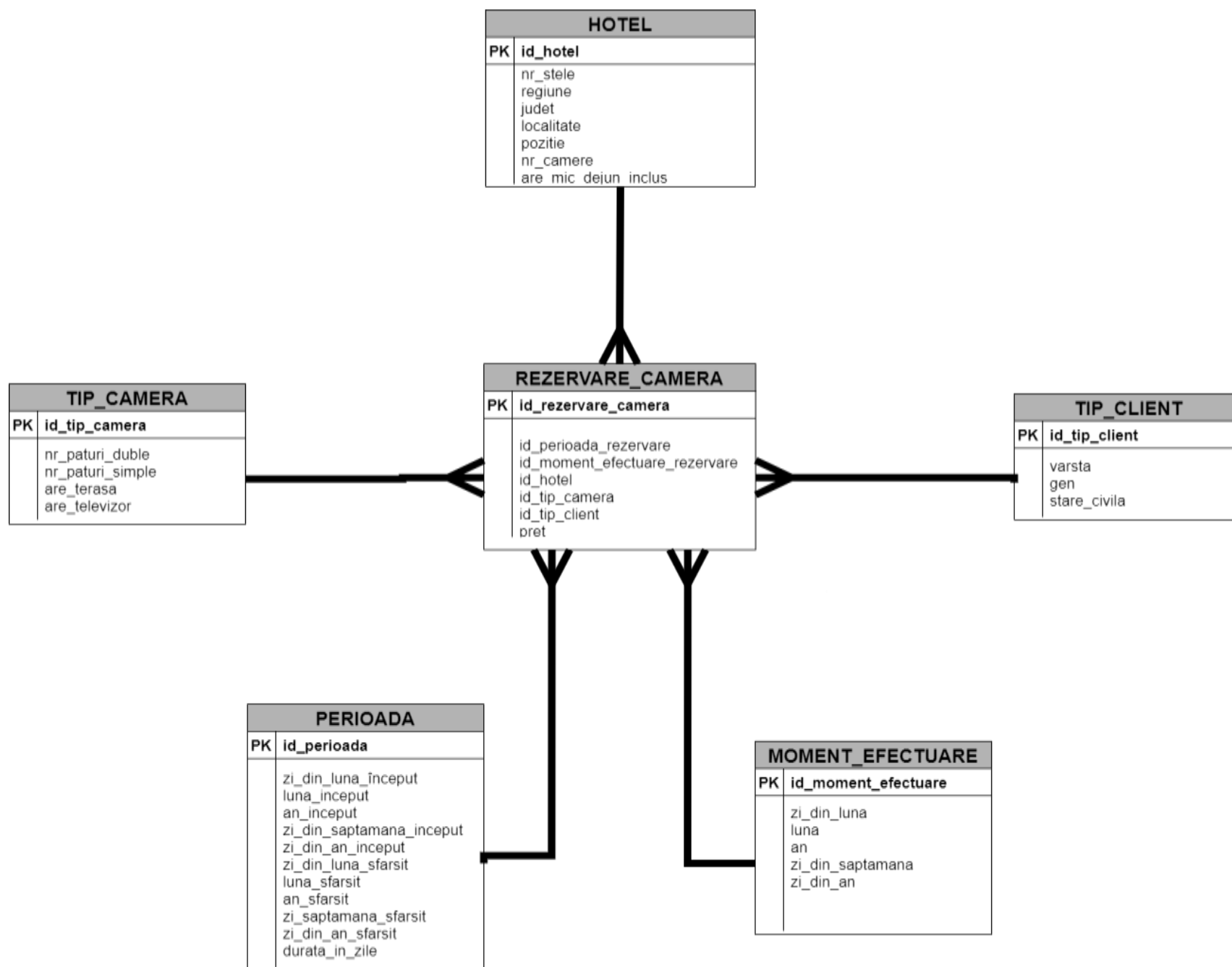
REZERVARE CAMERA (id_rezervare, id_camera).

CAMERA (#id_camera, id_hotel, nr_camera, nr_etaj, nr_paturi_duble, nr_paturi_simple, are_terasa, are_televizor, pret_per_noapte)

HOTEL (#id_hotel, id_zona, nume, nr_stele, are_mic_dejun)

ZONA (#id_zona, regiune, judet, localitate, pozitie).

3. Diagrama stea/fulg a bazei de date depozit (un tabel de fapte și cel puțin 5 tabele dimensiune).



4. (1p) Descrierea câmpurilor necesare pentru fiecare tabel din baza de date depozit și modul de populare al acestora cu informații din baza de date OLTP

HOTEL (#id_hotel, nr_stele, judet, localitate, pozitie, nr_camere, are_mic_dejun_inclus)

id_hotel - cheie primara autogenerata

nr_stele - se va prelua din tabela hotel a bazei de date OLTP

judet - se va utiliza join intre tabela hotel din OLTP si tabela locatie din OLTP

localitate- se va utiliza join intre tabela hotel din OLTP si tabela locatie din OLTP

pozitie - se va utiliza join intre tabela hotel din OLTP si tabela locatie din OLTP

nr_camere - se va prelua din tabela hotel a bazei de date OLTP

are_mic_dejun_inclus - se va prelua din tabela hotel a bazei de date OLTP

TIP CAMERA (#id_tip_camera, nr_paturi_duble, nr_paturi_simple, are_terasa,are_televizor)

id_tip_camera - cheie primara autogenerata

nr_paturi_duble - se va prelua din tabela camera a bazei de date OLTP

nr_paturi_simple - se va prelua din tabela camera a bazei de date OLTP

are_terasa- se va prelua din tabela camera a bazei de date OLTP

are_televizor- se va prelua din tabela camera a bazei de date OLTP

PERIOADA (#id_perioada, zi_luna_inceput, luna_inceput, an_inceput, zi_din_saptamana_inceput, zi_din_an_inceput, zi_luna_sfarsit, luna_sfarsit, an_sfarsit, zi_din_saptamana_sfarsit, zi_din_an_sfarsit, durata_in_zile)

id_perioada - cheie primara autogenerata

luna_inceput - se va prelua din tabela rezervare din OLTP

an_inceput - se va prelua din tabela rezervare din OLTP

zi_din_saptamana_inceput - se va prelua din tabela rezervare din OLTP

zi_din_an_inceput - se va prelua din tabela rezervare din OLTP

zi_luna_sfarsit - se va prelua din tabela rezervare din OLTP

luna_sfarsit - se va prelua din tabela rezervare din OLTP

an_sfarsit - se va prelua din tabela rezervare din OLTP

zi_din_saptamana_sfarsit - se va prelua din tabela rezervare din OLTP

zi_din_an_sfarsit - se va prelua din tabela rezervare din OLTP

durata_in_zile - se va prelua din tabela rezervare din OLTP

MOMENT EFECTUARE (#id_moment_efectuaire, zi_luna, luna, an, zi_din_saptamana, zi_din_an)

id_moment_efectuaire - cheie primara autogenerata

zi_luna-

luna-

an -

zi_din_saptamana-

TIP CLIENT (#id_tip_client, varsta, gen, stare_civila)

id_tip_client -

varsta - diferenta intre sysdate si data nasterii preluata din tabela client OLTP

gen - preluat din tabela client OLTP

stare_civila - preluat din tabela client OLTP

Pentru popularea bazei de date depozit cu date din baza de date OLTP se va utiliza o procedură. Au fost create funcții ajutătoare pentru obținerea mai multor informații, precum calculul vârstei clientului.

5. (1p) Identificarea constrângerilor specifice depozitelor de date ce trebuie definite, justificând alegerea făcută

- Pentru toate id-urile prezente în tabele se va aplica constrângerea de cheie primară (valoare unică și nenulă pentru fiecare tuplu).
- Pentru toate datele introduse în tabela “perioada” se vor aplica restricții de tipul NOT NULL, întrucât o rezervare fără date de început și final nu poate fi considerată validă.
- Pentru toate datele introduse în tabela “hotel” se vor aplica constrângeri de tipul NOT NULL. Vor fi necesare toate datele solicitate pentru a se considera un hotel valid. Aceste date sunt utile atât pentru clienți cât și pentru analiștii de date.
- Pentru ca un client să fie considerat valid, este necesar ca toate câmpurile tabelii asociate să aibă restricția NOT NULL. Aceasta este importantă și pentru a evita rezervările false.
- Pentru câmpurile unde este necesară introducerea datelor de tip zi din lună, se poate impune restricția ca valoarea acestora să fie cuprinsă între 1 și 31.
- Pentru câmpurile unde este necesară introducerea datelor de tip zi din an, se poate impune restricția ca valoarea acestora să fie cuprinsă între 1 și 365.

6. (0,5p) Identificarea indecșilor **specifci depozitelor de date** ce trebuie definiți asupra modelului (**minim 2** dacă echipa este formată din 4 persoane); formularea unei cereri în limbaj natural care va determina utilizarea indecșilor specificați și va fi implementată în următoarea etapă.

- **Indexare după ID-ul rezervării**

Întrucât rezervarea este pionul central al bazei de date, este util să definim un astfel de index pentru a accesa mai ușor detaliile unei rezervări. De asemenea, vrem să vedem câte camere au fost rezervate simultan de către același client. Poate fi de ajutor pentru raportare. Câmpul `id_rezervare` este preluat din tabelul “rezervare camera” din baza de date OLTP și astfel putem identifica ce camere au fost rezervate în cadrul acestui ID.

- **Indexare după luna efectuării rezervării**

Acest index poate fi utilizat pentru un raport în care se dorește observarea lunii din an în care se efectuează rezervările. Spre exemplu, o agenție de turism ar avea nevoie de astfel de date pentru a ști când se pot lansa oferte noi.

- **Indexare după localitatea în care se află hotelul și poziția acestuia față de centrul localității**

Acest index va fi construit pentru a putea sorta și grupa rezervările după locație. Locația are rol semnificativ în rapoartele și statisticile destinațiilor de vacanță preferate de către clienți.

- **Indexare după tip camera**

Acest index este construit pentru a sorta rezervările în funcție de tipul de cameră ales. Poate fi utilizat pentru a vedea mai ușor ce tip de cameră preferă clienții.

7. (0,5p) Identificarea obiectelor de tip dimensiune ce trebuie definite asupra modelului

- **Obiect de tip dimensiune pentru tabelul perioada rezervare**

Această dimensiune evidențiază dependențele dintr-zi zi (_inceput/_sfarsit) - zi_luna (_inceput/_sfarsit) - zi_an (_inceput/_sfarsit). Mai exact, ziua în care începe (/se sfârșește) rezervarea poate determina valorile atributelor zi din lună și zi din an.

- **Obiect de tip dimensiune pentru locație**

Am putea considera ierarhia localitate județ. Totuși, această dimensiune este problematică deoarece același nume de localitate se poate regăsi în mai multe județe. Este un caz bun pentru obiecte de tip dimensiune dar utilizat în raportări poate duce la erori grave.

- **Obiect de tip dimensiune pentru id rezervare**

Atributul id_rezervare determină attributele id_perioada_rezervare, id_moment_efectuare_rezervare, id_hotel, id_tip_client.

8. (1p) Identificarea tabelelor care vor fi partiționate și a tipului de partiționare. Formularea unei cereri în limbaj natural care va determina utilizarea lor și va fi implementată în următoarea etapă.

- **Partitionare prin range pret pentru tabelul rezervare_camera**

Se pot împărți rezervările în funcție de un interval de preț. Aceste intervale pot fi utilizate în raportările pentru bugetele clienților sau raportările pentru stabilirea ofertelor în piață.

- **Partitionare prin listă după numărul de stele pentru tabelul hotel**

Acest tip de partiționare se poate utiliza pentru situațiile în care se dorește obținerea unor informații în funcție de tipul hotelului indicat de numărul de stele. Spre exemplu, “Care este pretul mediu al unei camere cu un anumit numar de locuri la un hotel cu x nr de stele?”.

9. (0,5p) Formularea în limbaj natural a unei cereri SQL complexe care va fi optimizată în următoarea etapă, folosind tehnici specifice bazelor de date depozit. Precizarea tehnicilor de optimizare ce ar putea fi utilizate pentru această cerere particulară (avantaje / dezavantaje de utilizare pentru o anumită tehnică)

Construiți o cerere care să afișeze numele hotelurilor ce au valoarea rezervărilor cuprinse între două valori.

10. (2p) Formularea în limbaj natural a cel puțin 5 cereri cu grad de complexitate diferit, concretizate în rapoarte (grafice) ce vor fi create în următoarele etape

- Realizați un raport care arată numărul de camere rezervate în fiecare lună.
- Raport grafic care arată câte rezervări au fost făcute în anumite perioade ale anului.
- Raport grafic ce arată cele mai scumpe 5 hoteluri din România.

PROIECT DATA WAREHOUSE

~ Modul implementare baza de date ~

1. Crearea bazei de date OLTP și a utilizatorilor

--Baza noastra de date va avea 3 tipuri de utilizatori in aplicatie

- utilizatorul manager care va introduce schema si cu toate datele din aplicatie.
- acesta va avea acces la toate datele din schema oltp prin SELECT,UPDATE,DELETE,INSERT
- Managerul are posibilitatea de a introduce rezervari,modifica utilizatori, update pe rezervari.
- Practic are acces sa faca tot ce vrea pe schema
- acesta va avea acces si in olap pentru a vizualiza rapoartele scoase dar
- si de a modifica datele din tabele dupa propriul interes.

```
show con_name;  
alter session set container= orclpdb;  
show con_name;  
ALTER PLUGGABLE DATABASE orclpdb open;
```

```
CREATE USER dw_manager IDENTIFIED BY mng_pass;  
GRANT CREATE SESSION TO dw_manager;  
GRANT CREATE ANY TABLE TO dw_manager;  
GRANT CREATE ANY INDEX TO dw_manager;  
GRANT CREATE VIEW TO dw_manager;  
GRANT CREATE TRIGGER TO dw_manager;  
GRANT SELECT ANY TABLE TO dw_manager;  
GRANT DELETE ANY TABLE TO dw_manager;  
GRANT UPDATE ANY TABLE TO dw_manager;  
GRANT ALTER ANY TABLE TO dw_manager;  
GRANT UNLIMITED TABLESPACE TO dw_manager;  
--Pentru a vizualiza privilegiile adaugate putem folosi aceasta cerere asupra utlizatorului creat.  
SELECT *  
FROM session_privs;  
-- intro schema si datele furnizate pentru schema  
--SCHEMA CU TABELE TREBUIE RULATE IN dw_manager
```


--script creare schema oltp.txt
--DATELE DIN TABELE TREBUIE INTRODUSE IN dw_manager
--script inserare date oltp.txt

-- urmatorul tip va fi de tip admin care va avea posibilitatea de UPDATE peste toate tabele din schema OLTP
-- fara a avea posibilitatea de a sterge orice tip de inregistrare deoarece si inregistrarile neconforme pot reprezenta
-- un interes pentru manager.

```
CREATE USER dw_admin IDENTIFIED BY admin_pass;  
GRANT CREATE SESSION TO dw_admin;
```

```
GRANT SELECT ANY TABLE TO dw_admin;  
GRANT DELETE ANY TABLE TO dw_admin;  
GRANT UPDATE ANY TABLE TO dw_admin;  
GRANT ALTER ANY TABLE TO dw_admin;
```

--daca dorim sa oferim doar anumite privilegii mai restrictive asupra anumitor tabele putem folosi comanda urmatoare

```
--GRANT UPDATE ON dw_manager.rezervare TO dw_admin;  
--sau daca dorim sa nu mai folosim anumite privilegii precum cel de mai putem folosi comanda  
--REVOKE DELETE ON dw_manager.rezervare FROM dw_admin;
```

User DW_ADMIN created.

--pentru a accesa un tabel trebuie sa folosim dw_manager.nume_tabel deoarece altfel nu merge

Grant succeeded.

-- iar ultimul tip de utilizator este cel de utilizator care are
-- posibilitatea sa vizualizeze hotelurile si sa introduca date in rezervari.
-- acesta nu avea acces la baza de date.

Grant succeeded.

--ca si SYS putem rula urmatoarea cerere pentru a vizualiza care sunt care sunt privilegiile oferite
SELECT substr(grantee,1,20) grantee, owner,substr(table_name,1,15) table_name, grantor, privilege

Grant succeeded.

```
FROM DBA_TAB_PRIVS  
WHERE grantee like 'DW_%';
```

--cu aceasta comanda putem vedea doar privilegiile mai restrictive.

Grant succeeded.

Grant succeeded.

```

SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET ECHO OFF
REM *****
REM Create the UTILIZATOR table to hold users information for application
Prompt ***** Creating UTILIZATOR table ....

```

```

CREATE TABLE utilizator
( id_utilizator NUMBER GENERATED ALWAYS as IDENTITY(START WITH 1 INCREMENT BY 1),
  nume_utilizator VARCHAR(30) CONSTRAINT nume_utilizator_nn NOT NULL,
  hash_parola VARCHAR(25) CONSTRAINT hash_parola_utilizator_nn NOT NULL,
  nume_complet VARCHAR(30) CONSTRAINT nume_complet_utilizator_nn NOT NULL,
  telefon VARCHAR(15) CONSTRAINT telefon_utilizator_nn NOT NULL,
  email VARCHAR(50) CONSTRAINT email_utilizator_nn NOT NULL,
  data_nasterii DATE CONSTRAINT data_nasterii_utilizator_nn NOT NULL,
  gen VARCHAR(20) DEFAULT NULL,
  stare_civila VARCHAR(20) DEFAULT NULL);

```

```

CREATE UNIQUE INDEX id_utilizator_index
ON utilizator (id_utilizator);

```

```

ALTER TABLE utilizator
ADD ( CONSTRAINT id_nume_utilizator_pk PRIMARY KEY (id_utilizator)) ;

```

```

CREATE UNIQUE INDEX id_utilizator_index
ON utilizator (id_utilizator);

```

```

ALTER TABLE utilizator
ADD ( CONSTRAINT id_nume_utilizator_pk PRIMARY KEY (id_utilizator)) ;

```

```

Script Output x
Task completed in 0,401 seconds

***** Creating UTILIZATOR table ....

Table UTILIZATOR created.

INDEX ID_UTILIZATOR_INDEX created.

Table UTILIZATOR altered.

```

REM *****

REM Create the REZERVARE table to hold information for reservation of users

Prompt ***** Creating REZERVARE table

CREATE TABLE rezervare

```
( id_rezervare NUMBER GENERATED ALWAYS as IDENTITY(START WITH 1 INCREMENT BY 1),  
  id_client NUMBER CONSTRAINT id_client_utilizator_nn NOT NULL,  
  data_inceput DATE CONSTRAINT data_inceput_rezervare_nn NOT NULL,  
  data_sfarsit DATE CONSTRAINT data_sfarsit_rezervare_nn NOT NULL,  
  data_efectuarii DATE);
```

CREATE UNIQUE INDEX id_rezervare_index
ON rezervare (id_rezervare);

ALTER TABLE rezervare

ADD (CONSTRAINT id_rezervare_pk PRIMARY KEY (id_rezervare)) ;

REM *****

REM Create the REZERVARE CAMERA table to hold information about rooms of users

Prompt ***** Creating REZERVARE CAMERA table

CREATE TABLE rezervare_camera

```
( id_rezervare NUMBER CONSTRAINT id_rezervare_atribuie_nn NOT NULL,  
  id_camera NUMBER CONSTRAINT id_camera_atribuie_nn NOT NULL);
```

CREATE UNIQUE INDEX id_rezervare_camera_index
ON rezervare_camera (id_rezervare,id_camera);

ALTER TABLE rezervare_camera

ADD (CONSTRAINT id_rezervare_camera__pk PRIMARY KEY
(id_rezervare,id_camera));

```
***** Creating REZERVARE table ....  
  
Table REZERVARE created.  
  
INDEX ID_REZERVARE_INDEX created.  
  
Table REZERVARE altered.
```

```
***** Creating REZERVARE CAMERA table ....  
  
Table REZERVARE_CAMERA created.  
  
INDEX ID_REZERVARE_CAMERA_INDEX created.  
  
Table REZERVARE_CAMERA altered.
```

REM *****

REM Create the CAMERA table to hold informations about rooms

Prompt ***** Creating CAMERA table

CREATE TABLE camera

```
( id_camera NUMBER GENERATED ALWAYS as IDENTITY(START WITH 1 INCREMENT BY 1),
  id_hotel NUMBER CONSTRAINT id_hotel_camera_nn NOT NULL,
  nr_camera NUMBER,
  nr_etaj NUMBER,
  nr_paturi_duble NUMBER CONSTRAINT nr_paturi_duble_camera_nn NOT NULL,
  nr_paturi_simple NUMBER CONSTRAINT nr_paturi_simple_camera_nn NOT NULL,
  are_terasa NUMBER(1) CONSTRAINT are_terasa_camera_nn NOT NULL,
  are_televizor NUMBER(1) CONSTRAINT are_televizor_camera_nn NOT NULL,
  pret_per_noapte NUMBER CONSTRAINT pret_per_noapte_camera_nn NOT NULL);
```

CREATE UNIQUE INDEX id_camera_index
ON camera (id_camera);

ALTER TABLE camera

ADD (CONSTRAINT id_camera_camera_pk PRIMARY KEY (id_camera)) ;

REM *****

REM Create the HOTEL table to hold information of hotels

Prompt ***** Creating HOTEL table

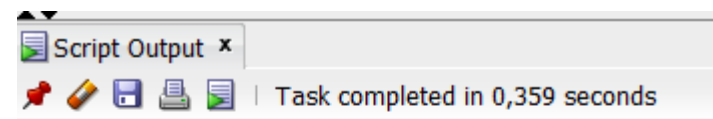
CREATE TABLE hotel

```
( id_hotel NUMBER GENERATED ALWAYS as IDENTITY(START WITH 1  
INCREMENT BY 1),
```

```
  nume VARCHAR2(50) CONSTRAINT nume_hotel_hotel_nn NOT NULL,
  nr_stele NUMBER CONSTRAINT nr_stele_nn NOT NULL,
  id_zona NUMBER CONSTRAINT id_zona_hotel_nn NOT NULL,
  are_mic_dejun_inclus NUMBER(1) CONSTRAINT are_mic_dejun_inclus_hotel_nn NOT NULL);
```

CREATE UNIQUE INDEX id_hotel_index ON hotel (id_hotel);

ALTER TABLE hotel ADD (CONSTRAINT id_hotel_hotel_pk PRIMARY KEY (id_hotel)) ;

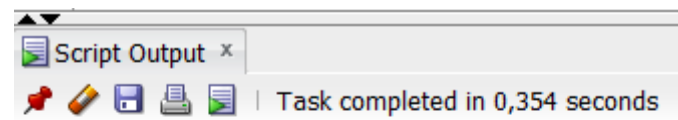


***** Creating CAMERA table

Table CAMERA created.

INDEX ID_CAMERA_INDEX created.

Table CAMERA altered.



***** Creating HOTEL table

Table HOTEL created.

INDEX ID_HOTEL_INDEX created.

Table HOTEL altered.

REM *****

REM Create the ZONA table to hold information the zones where the hotels was build

Prompt ***** Creating ZONA table

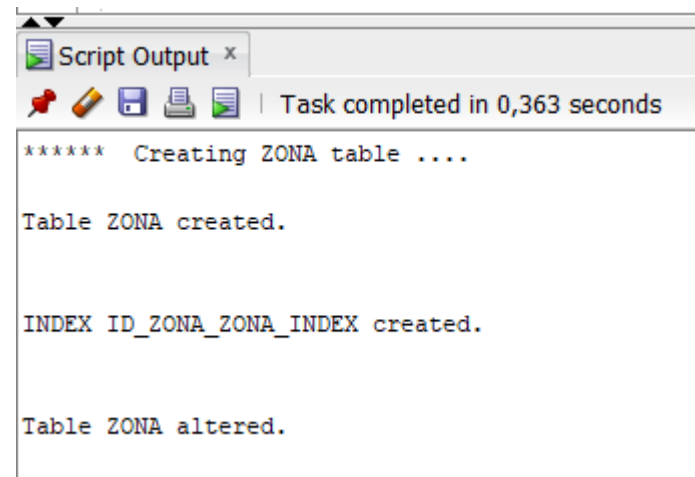
CREATE TABLE zona

(id_zona NUMBER GENERATED ALWAYS as IDENTITY(START WITH 1 INCREMENT BY 1),
regiune VARCHAR2(50),
judet VARCHAR2(50) CONSTRAINT judet_zona_nn NOT NULL,
localitate VARCHAR2(50) CONSTRAINT localitate_zona_nn NOT NULL,
pozitie VARCHAR2(50) CONSTRAINT pozitie_zona_nn NOT NULL);

CREATE UNIQUE INDEX id_zona_zona_index
ON zona (id_zona);

ALTER TABLE zona

ADD (CONSTRAINT id_zona_zona_pk PRIMARY KEY (id_zona)) ;



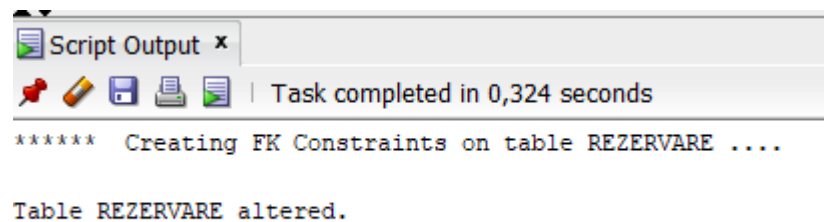
REM *****Introducerea de FK tabelului*****

REZERVARE*****

Prompt ***** Creating FK Constraints on table REZERVARE

ALTER TABLE rezervare

add constraint fk_id_client_id_utilizator FOREIGN KEY(id_client) REFERENCES utilizator(id_utilizator);



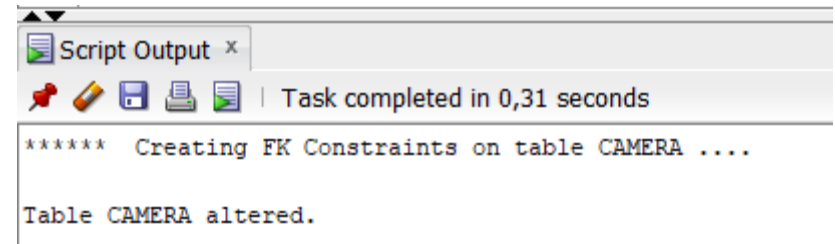
REM *****Introducerea de FK tabelului*****

CAMERA*****

Prompt ***** Creating FK Constraints on table CAMERA

ALTER TABLE camera

add constraint fk_camera_hotel FOREIGN KEY(id_hotel) REFERENCES
hotel(id_hotel);



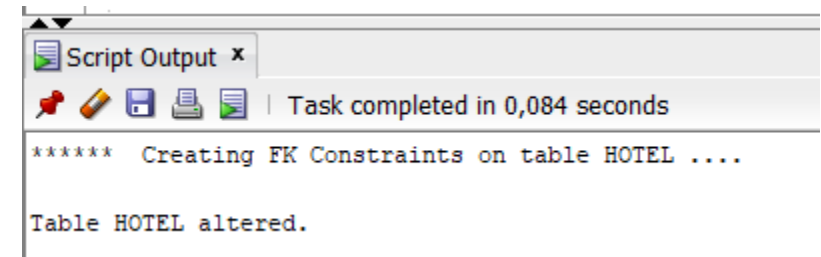
REM *****Introducerea de FK tabelului*****

HOTEL*****

Prompt ***** Creating FK Constraints on table HOTEL

ALTER TABLE hotel

add constraint fk_hotel_zona FOREIGN KEY(id_zona) REFERENCES
zona(id_zona);



REM *****Introducerea de FK tabelului***** REZERVARE_CAMERA*****

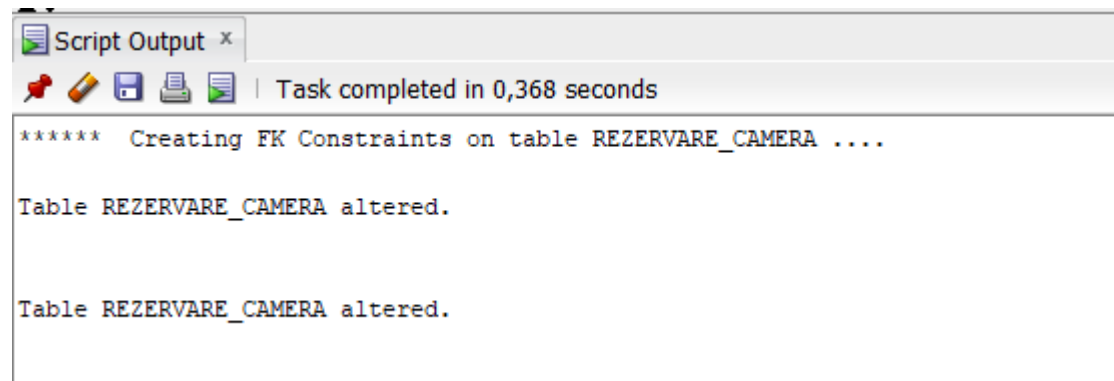
Prompt ***** Creating FK Constraints on table REZERVARE_CAMERA

ALTER TABLE rezervare_camera

ADD CONSTRAINT fk_rezervare_camera_rezervare FOREIGN KEY(id_rezervare) REFERENCES rezervare(id_rezervare);

ALTER TABLE rezervare_camera

ADD CONSTRAINT fk_rezervare_camera_camera FOREIGN KEY (id_camera) REFERENCES camera(id_camera);



2. (0,25p) Generarea datelor și inserarea acestora în tabele.

---Inserarea datelor in tabelul ZONA---

```
INSERT INTO ZONA (JUDET, LOCALITATE, POZITIE) VALUES ('Mures','Targu Mures','centrala'); COMMIT;
INSERT INTO ZONA (JUDET, LOCALITATE, POZITIE) VALUES ('Brasov','Bran','centrala'); COMMIT;
INSERT INTO ZONA (JUDET, LOCALITATE, POZITIE) VALUES ('Brasov','Bran','periferica'); COMMIT;
INSERT INTO ZONA (JUDET, LOCALITATE, POZITIE) VALUES ('Bucuresti','Bucuresti','centrala'); COMMIT;
INSERT INTO ZONA (JUDET, LOCALITATE, POZITIE) VALUES ('Prahova','Sinaia','centrala'); COMMIT;
```



---Inserarea datelor in tabelul HOTEL(dependent de ZONA)---

```
INSERT INTO HOTEL (NUME, NR_STELE, ID_ZONA, ARE_MIC_DEJUN_INCLUS) VALUES ('Hotel Privo',4,1,0); COMMIT;
INSERT INTO HOTEL (NUME, NR_STELE, ID_ZONA, ARE_MIC_DEJUN_INCLUS) VALUES ('Conacul Bratescu', 4, 2, 1);
COMMIT;
INSERT INTO HOTEL (NUME, NR_STELE, ID_ZONA, ARE_MIC_DEJUN_INCLUS) VALUES ('Transylvanian Inn', 3, 3, 1);
COMMIT;
INSERT INTO HOTEL (NUME, NR_STELE, ID_ZONA, ARE_MIC_DEJUN_INCLUS) VALUES ('The Mansion Boutique Hotel', 4, 4,
1); COMMIT;
INSERT INTO HOTEL (NUME, NR_STELE, ID_ZONA, ARE_MIC_DEJUN_INCLUS) VALUES ('Ioana Hotel', 5, 5, 1); COMMIT;
```

Worksheet Query Builder

---Inserarea datelor in tabelul HOTEL(dependent de ZONA)---

```
INSERT INTO HOTEL (NUME, NR_STELE, ID_ZONA, ARE_MIC_DEJUN_INCLUS) VALUES ('Hotel Privo',4,1,0);
INSERT INTO HOTEL (NUME, NR_STELE, ID_ZONA, ARE_MIC_DEJUN_INCLUS) VALUES ('Conacul Bratescu', 4, 2, 1);
INSERT INTO HOTEL (NUME, NR_STELE, ID_ZONA, ARE_MIC_DEJUN_INCLUS) VALUES ('Transylvanian Inn', 3, 3, 1);
INSERT INTO HOTEL (NUME, NR_STELE, ID_ZONA, ARE_MIC_DEJUN_INCLUS) VALUES ('The Mansion Boutique Hotel', 4, 4, 1);
INSERT INTO HOTEL (NUME, NR_STELE, ID_ZONA, ARE_MIC_DEJUN_INCLUS) VALUES ('Ioana Hotel', 5, 5, 1);
INSERT INTO HOTEL (NUME, NR_STELE, ID_ZONA, ARE_MIC_DEJUN_INCLUS) VALUES ('Epoque Hotel Relais Chateaux', 5, 6, 0);
INSERT INTO HOTEL (NUME, NR_STELE, ID_ZONA, ARE_MIC_DEJUN_INCLUS) VALUES ('Rosen Villa Sibiu', 1, 7, 1);
INSERT INTO HOTEL (NUME, NR_STELE, ID_ZONA, ARE_MIC_DEJUN_INCLUS) VALUES ('Hotel International Iasi', 4, 8, 1);
```

Script Output x

Task completed in 0,83 seconds

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

---Inserarea datelor in tabelul CAMERA(dependent de HOTEL)---

```
INSERT INTO CAMERA (ID_HOTEL, NR_CAMERA, NR_ETAJ, NR_PATURI_DUBLE, NR_PATURI_SIMPLE, ARE_TERASA,
ARE_TELEVIZOR, PRET_PER_NOAPTE)
VALUES (76, 312, 3, 0, 3, 1, 0, 450); COMMIT;
```

```
INSERT INTO CAMERA (ID_HOTEL, NR_CAMERA, NR_ETAJ, NR_PATURI_DUBLE, NR_PATURI_SIMPLE, ARE_TERASA,
ARE_TELEVIZOR, PRET_PER_NOAPTE)
VALUES (78, 107, 1, 0, 1, 1, 0, 100); COMMIT;
```

```
INSERT INTO CAMERA (ID_HOTEL, NR_CAMERA, NR_ETAJ, NR_PATURI_DUBLE, NR_PATURI_SIMPLE, ARE_TERASA,
ARE_TELEVIZOR, PRET_PER_NOAPTE)
VALUES (44, 405, 4, 0, 2, 1, 0, 180); COMMIT;
```


Worksheet
Query Builder

```

---Inserarea datelor in tabelul CAMERA(dependent de HOTEL)---

INSERT INTO CAMERA (ID_HOTEL, NR_CAMERA, NR_ETAJ, NR_PATURI_DUBLE, NR_PATURI_SIMPLE, ARE_TERASA, ARE_TELEVIZOR, PRET_PER_NOAPTE)
VALUES (76, 312, 3, 0, 3, 1, 0, 450);

INSERT INTO CAMERA (ID_HOTEL, NR_CAMERA, NR_ETAJ, NR_PATURI_DUBLE, NR_PATURI_SIMPLE, ARE_TERASA, ARE_TELEVIZOR, PRET_PER_NOAPTE)
VALUES (78, 107, 1, 0, 1, 1, 0, 100);

INSERT INTO CAMERA (ID_HOTEL, NR_CAMERA, NR_ETAJ, NR_PATURI_DUBLE, NR_PATURI_SIMPLE, ARE_TERASA, ARE_TELEVIZOR, PRET_PER_NOAPTE)
VALUES (44, 405, 4, 0, 2, 1, 0, 180);

```

Script Output x
Task completed in 1,639 seconds

```

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

```

---Inserarea datelor in tabelul UTILIZATOR---

```

INSERT INTO UTILIZATOR (NUME_UTILIZATOR, HASH_PAROLA, NUME_COMPLET, TELEFON, EMAIL, DATA_NASTERII, GEN, STARE_CIVILA)
VALUES ('bernard_noble', 'kr102h8', 'Bernard Noble', '+40 710 024 027', 'kstoltenberg@yahoo.com', to_date('13-Oct-1976', 'DD-MON-RR'), 'masculin', 'necasatorit '); COMMIT;

```

```

INSERT INTO UTILIZATOR (NUME_UTILIZATOR, HASH_PAROLA, NUME_COMPLET, TELEFON, EMAIL, DATA_NASTERII, GEN, STARE_CIVILA)
VALUES ('isis_saunders', 'PA3pnYV', 'Isis Saunders', '+40 713 721 929', 'orn.skye@huels.com', to_date('6-Apr-2001', 'DD-MON-RR'), 'feminin', 'casatorita'); COMMIT;

```

```

INSERT INTO UTILIZATOR (NUME_UTILIZATOR, HASH_PAROLA, NUME_COMPLET, TELEFON, EMAIL, DATA_NASTERII, GEN, STARE_CIVILA)
VALUES ('micaela_gillespie', 'G7fSXy8', 'Micaela Gillespie', '+40 713 037 240', 'stuart22@yahoo.com', to_date('19-Apr-1966', 'DD-MON-RR'), 'feminin', 'necasatorita'); COMMIT;

```

```

INSERT INTO UTILIZATOR (NUME_UTILIZATOR, HASH_PAROLA, NUME_COMPLET, TELEFON, EMAIL, DATA_NASTERII, GEN, STARE_CIVILA)
VALUES ('gilbert_mccarty', 'ek1Rwcq', 'Gilbert Mccarty', '+40 711 666 147', 'gskiles@altenwerth.com', to_date('31-Jul-1969', 'DD-MON-RR'), 'masculin', 'necasatorit '); COMMIT;

```

Worksheet Query Builder

---Inserarea datelor in tabelul UTILIZATOR---

```
INSERT INTO UTILIZATOR (NUME_UTILIZATOR, HASH_PAROLA, NUME_COMPLET, TELEFON, EMAIL, DATA_NASTERII, GEN, STARE_CIVILA)
VALUES ('bernard_noble', 'kr1O2h8', 'Bernard Noble', '+40 710 024 027', 'kstoltenberg@yahoo.com', to_date('13-Oct-1976', 'DD-MON-RR'), 'male', 'necasatorit ');

INSERT INTO UTILIZATOR (NUME_UTILIZATOR, HASH_PAROLA, NUME_COMPLET, TELEFON, EMAIL, DATA_NASTERII, GEN, STARE_CIVILA)
VALUES ('isis_saunders', 'PA3pnYV', 'Isis Saunders', '+40 713 721 929', 'orn.skye@huelis.com', to_date('6-Apr-2001', 'DD-MON-RR'), 'female', 'casatorita');

INSERT INTO UTILIZATOR (NUME_UTILIZATOR, HASH_PAROLA, NUME_COMPLET, TELEFON, EMAIL, DATA_NASTERII, GEN, STARE_CIVILA)
```

Script Output x

Task completed in 0,768 seconds

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

```
---Inserarea datelor in tabelul REZERVARE(dependent de UTILIZATOR)---
INSERT INTO REZERVARE (ID_CLIENT, DATA_INCEPUT, DATA_SFARSIT)
VALUES (89, to_date('2-Jul-2023', 'DD-MON-RR'), to_date('9-Jul-2023', 'DD-MON-RR')); COMMIT;

INSERT INTO REZERVARE (ID_CLIENT, DATA_INCEPUT, DATA_SFARSIT)
VALUES (49, to_date('4-Feb-2025', 'DD-MON-RR'), to_date('13-Feb-2025', 'DD-MON-RR')); COMMIT;

INSERT INTO REZERVARE (ID_CLIENT, DATA_INCEPUT, DATA_SFARSIT)
VALUES (4, to_date('24-Apr-2023', 'DD-MON-RR'), to_date('27-Apr-2023', 'DD-MON-RR')); COMMIT;
```

Worksheet Query Builder

---Inserarea datelor in tabelul REZERVARE(dependent de UTILIZATOR)---

```
INSERT INTO REZERVARE (ID_CLIENT, DATA_INCEPUT, DATA_SFARSIT)
VALUES (89, to_date('2-Jul-2023', 'DD-MON-RR'), to_date('9-Jul-2023', 'DD-MON-RR'));

INSERT INTO REZERVARE (ID_CLIENT, DATA_INCEPUT, DATA_SFARSIT)
VALUES (49, to_date('4-Feb-2025', 'DD-MON-RR'), to_date('13-Feb-2025', 'DD-MON-RR'));

INSERT INTO REZERVARE (ID_CLIENT, DATA_INCEPUT, DATA_SFARSIT)
VALUES (4, to_date('24-Apr-2023', 'DD-MON-RR'), to_date('27-Apr-2023', 'DD-MON-RR'));
```

Script Output x

Task completed in 0,735 seconds

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

---Inserarea datelor in tabelul REZERVARE_CAMERA---

```
INSERT INTO REZERVARE_CAMERA (ID_REZERVARE, ID_CAMERA)
VALUES (95, 167);
```

```
INSERT INTO REZERVARE_CAMERA (ID_REZERVARE, ID_CAMERA)
VALUES (93, 54);
```

```
INSERT INTO REZERVARE_CAMERA (ID_REZERVARE, ID_CAMERA)
VALUES (47, 140);
```

```
INSERT INTO REZERVARE_CAMERA (ID_REZERVARE, ID_CAMERA)
VALUES (93, 71);
```

```
INSERT INTO REZERVARE_CAMERA (ID_REZERVARE, ID_CAMERA)
VALUES (10, 154);
```

```
INSERT INTO REZERVARE_CAMERA (ID_REZERVARE, ID_CAMERA)
VALUES (4, 120);
```

Worksheet Query Builder

```

=====
---Inserarea datelor in tabelul REZERVARE_CAMERA---

INSERT INTO REZERVARE_CAMERA (ID_REZERVARE, ID_CAMERA)
VALUES (55, 167);

INSERT INTO REZERVARE_CAMERA (ID_REZERVARE, ID_CAMERA)
VALUES (53, 54);

INSERT INTO REZERVARE_CAMERA (ID_REZERVARE, ID_CAMERA)
VALUES (47, 140);

INSERT INTO REZERVARE_CAMERA (ID_REZERVARE, ID_CAMERA)
VALUES (52, 71);

```

Script Output x

Task completed in 0,986 seconds

```

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

```

```

--GENEREAZA O DATA ALEATOARE A EFECTUARII PENTRU FIECARE REZERVARE
UPDATE rezervare SET data_efectuarii=TO_DATE(TRUNC(DBMS_RANDOM.VALUE(TO_CHAR(
DATE'2022-01-01','J'),TO_CHAR(
DATE'2022-12-31','J'))),'J'); COMMIT;

```

Worksheet Query Builder

```

--GENEREAZA O DATA ALEATOARE A EFECTUARII PENTRU FIECARE REZERVARE
UPDATE rezervare SET data_efectuarii=TO_DATE(TRUNC(DBMS_RANDOM.VALUE(TO_CHAR(
DATE'2022-01-01','J'),TO_CHAR(
DATE'2022-12-31','J'))),'J'); COMMIT;

```

Script Output x

Task completed in 0,272 seconds

```

99 rows updated.

Commit complete.

```

```
--SETEAZA REGIUNILE ASOCIATE JUDETELOR
```

```
UPDATE zona
```

```
SET regiune='Transilvania'
```

```
WHERE judet='Cluj' OR judet='Brasov' OR judet='Sibiu' OR judet='Mures'; COMMIT;
```

```
UPDATE zona
```

```
SET regiune='Maramures'
```

```
WHERE judet='Maramures' OR judet='Satu Mare'; COMMIT;
```

```
UPDATE zona
```

```
SET regiune='Muntenia'
```

```
WHERE judet='Prahova' OR judet='Ilfov' OR judet='Arges' OR judet='Buzau' OR judet='Bucuresti'; COMMIT;
```

```
UPDATE zona
```

```
SET regiune='Oltenia'
```

```
WHERE judet='Valcea'; COMMIT;
```

```
UPDATE zona
```

```
SET regiune='Dobrogea'
```

```
WHERE judet='Constanta'; COMMIT;
```

Worksheet Query Builder	
<pre>--SETEAZA REGIUNILE ASOCIATE JUDETELOR UPDATE zona SET regiune='Transilvania' WHERE judet='Cluj' OR judet='Brasov' OR judet='Sibiu' OR judet='Mures'; COMMIT; UPDATE zona SET regiune='Maramures' WHERE judet='Maramures' OR judet='Satu Mare'; COMMIT; UPDATE zona SET regiune='Muntenia' WHERE judet='Prahova' OR judet='Ilfov' OR judet='Arges' OR judet='Buzau' OR judet='Bucuresti'; COMMIT; UPDATE zona SET regiune='Oltenia' WHERE judet='Valcea'; COMMIT; UPDATE zona SET regiune='Dobrogea' WHERE judet='Constanta'; COMMIT;</pre>	
<div>Script Output x</div> <div>Task completed in 0,114 seconds</div>	
<pre>14 rows updated. Commit complete. 5 rows updated. Commit complete. 6 rows updated. Commit complete.</pre>	

3. (0,5p) Crearea bazei de date depozit și a utilizatorilor

Schema bazei de date depozit a fost introdusă în conexiunea utilizatorului dw_manager. Acest utilizator a fost creat asemănător cu utilizatorul pentru baza de date OLTP.

```
CREATE TABLE perioada_rezervare_OLAP(  
id_perioada NUMBER(8,0) GENERATED ALWAYS as IDENTITY(START WITH 1 INCREMENT BY 1),  
zi_din_luna_inceput NUMBER(2,0) CONSTRAINT zi_din_luna_inceput_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
luna_inceput CHAR(3) CONSTRAINT luna_inceput_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
an_inceput NUMBER(4,0) CONSTRAINT an_inceput_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
zi_din_saptamana_inceput CHAR(3) CONSTRAINT zi_din_saptamana_inceput_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
zi_din_an_inceput NUMBER(3,0) CONSTRAINT zi_din_an_inceput_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
zi_din_luna_sfarsit NUMBER(2,0) CONSTRAINT zi_din_luna_sfarsit_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
luna_sfarsit CHAR(3) CONSTRAINT luna_sfarsit_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
an_sfarsit NUMBER(4,0) CONSTRAINT an_sfarsit_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
zi_din_saptamana_sfarsit CHAR(3) CONSTRAINT zi_din_saptamana_sfarsit_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
zi_din_an_sfarsit NUMBER(3,0) CONSTRAINT zi_din_an_sfarsit_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
durata_in_zile NUMBER(2,0) CONSTRAINT durata_in_zile_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
PRIMARY KEY(id_perioada)  
);
```

The screenshot displays a database management interface with a 'Worksheet' tab and a 'Query Builder' tab. The 'Query Builder' tab is active, showing a SQL script for creating a table named 'perioada_rezervare_OLAP'. The script includes various constraints and a primary key. Below the script, a 'Script Output' window shows the message 'Table PERIODA_REZERVARE_OLAP created.' and a status bar indicates 'Task completed in 0,064 seconds'.

```
CREATE TABLE perioada_rezervare_OLAP(  
id_perioada NUMBER(8,0) GENERATED ALWAYS as IDENTITY(START WITH 1 INCREMENT BY 1),  
zi_din_luna_inceput NUMBER(2,0) CONSTRAINT zi_din_luna_inceput_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
luna_inceput CHAR(3) CONSTRAINT luna_inceput_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
an_inceput NUMBER(4,0) CONSTRAINT an_inceput_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
zi_din_saptamana_inceput CHAR(3) CONSTRAINT zi_din_saptamana_inceput_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
zi_din_an_inceput NUMBER(3,0) CONSTRAINT zi_din_an_inceput_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
zi_din_luna_sfarsit NUMBER(2,0) CONSTRAINT zi_din_luna_sfarsit_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
luna_sfarsit CHAR(3) CONSTRAINT luna_sfarsit_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
an_sfarsit NUMBER(4,0) CONSTRAINT an_sfarsit_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
zi_din_saptamana_sfarsit CHAR(3) CONSTRAINT zi_din_saptamana_sfarsit_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
zi_din_an_sfarsit NUMBER(3,0) CONSTRAINT zi_din_an_sfarsit_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
durata_in_zile NUMBER(2,0) CONSTRAINT durata_in_zile_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
PRIMARY KEY(id_perioada)  
);
```

Script Output x
Task completed in 0,064 seconds

Table PERIODA_REZERVARE_OLAP created.

```

CREATE TABLE tip_camera_OLAP(
    id_tip_camera NUMBER(8) GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1),
    nr_paturi_duble NUMBER(1) CONSTRAINT nr_paturi_duble_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
    nr_paturi_simple NUMBER(1) CONSTRAINT nr_paturi_simple_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
    are_terasa NUMBER(1) CONSTRAINT are_terasa_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
    are_televizor NUMBER(1) CONSTRAINT are_televizor_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
    PRIMARY KEY (id_tip_camera));

```

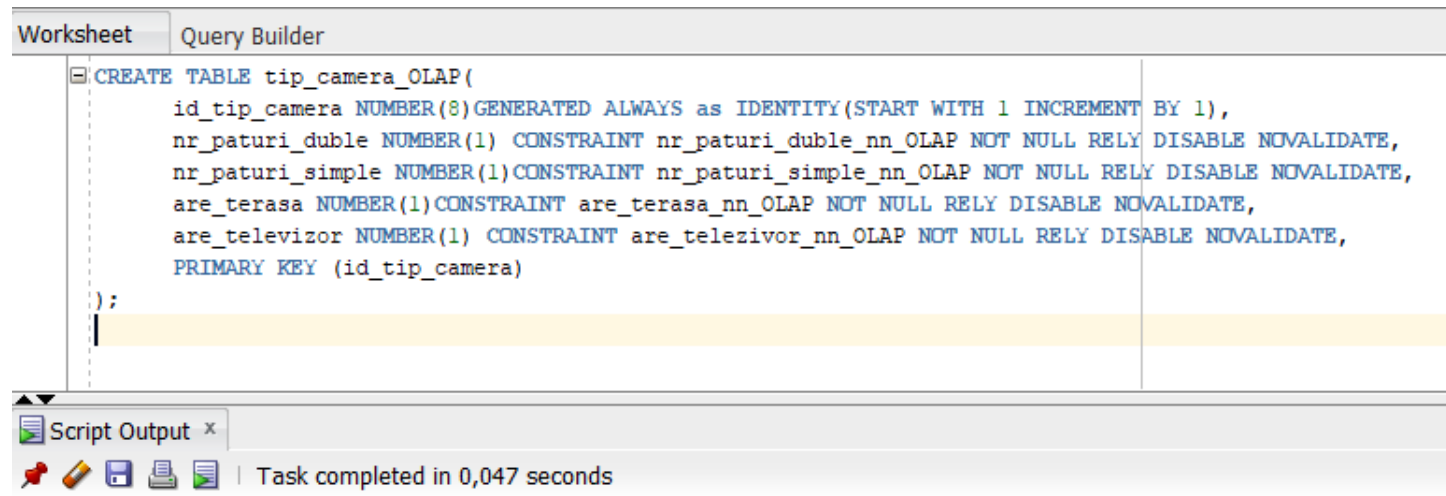


Table TIP_CAMERA_OLAP created.

```

CREATE TABLE hotel_OLAP
(
    id_hotel NUMBER(8),
    nume VARCHAR2(50) CONSTRAINT nume_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
    nr_stele NUMBER(1) CONSTRAINT nr_stele_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
    regiune VARCHAR2(50),
    judet VARCHAR2(20) CONSTRAINT judet_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
    localitate VARCHAR(20) CONSTRAINT localitate_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
    pozitie VARCHAR(20) CONSTRAINT pozitie_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
    nr_camere NUMBER(3),
    are_mic_dejun_inclus NUMBER(1) CONSTRAINT are_mic_dejun_inclus_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
    PRIMARY KEY (id_hotel))

```



```

PARTITION BY LIST (nr_stele)
(PARTITION o_stea VALUES (1),
PARTITION doua_stele VALUES (2),
PARTITION trei_stele VALUES (3),
PARTITION patru_stele VALUES (4),
PARTITION cinci_stele VALUES (5));

```

Worksheet	Query Builder
<pre> CREATE TABLE hotel_OLAP (id_hotel NUMBER(8), nume VARCHAR2(50) CONSTRAINT nume_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE, nr_stele NUMBER(1) CONSTRAINT nr_stele_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE, regiune VARCHAR2(50), judet VARCHAR2(20) CONSTRAINT judet_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE, localitate VARCHAR(20) CONSTRAINT localitate_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE, pozitie VARCHAR(20) CONSTRAINT pozitie_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE, nr_camere NUMBER(3), are_mic_dejun_inclus NUMBER(1) CONSTRAINT are_mic_dejun_inclus_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE, PRIMARY KEY (id_hotel)) PARTITION BY LIST (nr_stele) (PARTITION o_stea VALUES (1), PARTITION doua_stele VALUES (2), PARTITION trei_stele VALUES (3), PARTITION patru_stele VALUES (4), PARTITION cinci_stele VALUES (5)); </pre>	
<div> <div>Script Output x</div> <div> Task completed in 0,081 seconds </div> </div>	
<p>Table HOTEL_OLAP created.</p>	

```

CREATE TABLE tip_client_OLAP(
    id_tip_client NUMBER(8) GENERATED ALWAYS as IDENTITY(START WITH 1 INCREMENT BY 1),
    varsta NUMBER(3) CONSTRAINT varsta_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
    gen VARCHAR(20),
    stare_civila VARCHAR(20),
    PRIMARY KEY (id_tip_client)
);

```

Worksheet Query Builder






```
CREATE TABLE tip_client_OLAP(  
  id_tip_client NUMBER(8) GENERATED ALWAYS as IDENTITY(START WITH 1 INCREMENT BY 1),  
  varsta NUMBER(3) CONSTRAINT varsta_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
  gen VARCHAR(20),  
  stare_civila VARCHAR(20),  
  PRIMARY KEY (id_tip_client)  
);
```

Script Output x

Task completed in 0,07 seconds

Table TIP_CLIENT_OLAP created.

```
CREATE TABLE moment_efectuare_rezervare_OLAP(  
  id_moment_efectuare NUMBER(8,0) GENERATED ALWAYS as IDENTITY(START WITH 1 INCREMENT BY 1),  
  zi_din_luna NUMBER(2,0) CONSTRAINT zi_din_luna_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
  luna CHAR(3) CONSTRAINT luna_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
  an NUMBER(4,0) CONSTRAINT an_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
  zi_din_saptamana CHAR(3) CONSTRAINT zi_din_saptamana_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
  zi_din_an NUMBER(3,0) CONSTRAINT zi_din_an_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
  --ora_aprox NUMBER(2,0),  
  PRIMARY KEY (id_moment_efectuare)  
);
```

Worksheet	Query Builder
	<pre> CREATE TABLE moment_efectuare_rezervare_OLAP(id_moment_efectuare NUMBER(8,0) GENERATED ALWAYS as IDENTITY(START WITH 1 INCREMENT BY 1), zi_din_luna NUMBER(2,0) CONSTRAINT zi_din_luna_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE, luna CHAR(3) CONSTRAINT luna_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE, an NUMBER(4,0) CONSTRAINT an_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE, zi_din_saptamana CHAR(3) CONSTRAINT zi_din_saptamana_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE, zi_din_an NUMBER(3,0) CONSTRAINT zi_din_an_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE, --ora_aprox NUMBER(2,0), PRIMARY KEY (id_moment_efectuare)); </pre>
<div> <div>Script Output x</div> <div>      </div> <div>Task completed in 0,06 seconds</div> </div>	
<p>Table MOMENT_EFECTUARE_REZERVARE_OLAP created.</p>	

```

DROP TABLE rezervare_camera_OLAP;
CREATE TABLE rezervare_camera_OLAP(
id_rezervare NUMBER(8,0) CONSTRAINT id_rezervare_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
id_hotel NUMBER(8,0) CONSTRAINT id_hotel_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
id_perioada NUMBER(8,0) CONSTRAINT id_perioada_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
id_moment_efectuare NUMBER(8,0) CONSTRAINT id_moment_efectuare_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
id_tip_client NUMBER(8,0) CONSTRAINT id_tip_client_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
id_tip_camera NUMBER(8,0) CONSTRAINT id_tip_camera_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
pret NUMBER CONSTRAINT pret_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE
);

```

Worksheet Query Builder

```
CREATE TABLE rezervare_camera_OLAP(  
id_rezervare NUMBER(8,0) CONSTRAINT id_rezervare_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
id_hotel NUMBER(8,0) CONSTRAINT id_hotel_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
id_perioada NUMBER(8,0) CONSTRAINT id_perioada_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
id_moment_efectuare NUMBER(8,0) CONSTRAINT id_moment_efectuare_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
id_tip_client NUMBER(8,0) CONSTRAINT id_tip_client_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
id_tip_camera NUMBER(8,0) CONSTRAINT id_tip_camera_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,  
pret NUMBER CONSTRAINT pret_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE  
);
```

Script Output x

Task completed in 0,062 seconds

Table REZERVARE_CAMERA_OLAP created.

```
ALTER TABLE rezervare_camera_OLAP  
ADD FOREIGN KEY(id_hotel) REFERENCES hotel_OLAP(id_hotel);  
  
ALTER TABLE rezervare_camera_OLAP  
ADD FOREIGN KEY(id_perioada) REFERENCES perioada_rezervare_OLAP(id_perioada);  
  
ALTER TABLE rezervare_camera_OLAP  
ADD FOREIGN KEY(id_moment_efectuare) REFERENCES moment_efectuare_rezervare_OLAP(id_moment_efectuare);  
  
ALTER TABLE rezervare_camera_OLAP  
ADD FOREIGN KEY(id_tip_camera) REFERENCES tip_camera_OLAP(id_tip_camera);  
  
ALTER TABLE rezervare_camera_OLAP  
ADD FOREIGN KEY(id_tip_client) REFERENCES tip_client_OLAP(id_tip_client);
```

Worksheet

Query Builder

ALTER TABLE rezervare_camera_OLAP

ADD FOREIGN KEY(id_hotel) REFERENCES hotel_OLAP(id_hotel);

ALTER TABLE rezervare_camera_OLAP

ADD FOREIGN KEY(id_perioada) REFERENCES perioada_rezervare_OLAP(id_perioada);

ALTER TABLE rezervare_camera_OLAP

ADD FOREIGN KEY(id_moment_efectuare) REFERENCES moment_efectuare_rezervare_OLAP(id_moment_efectuare);






ALTER TABLE rezervare_camera_OLAP

ADD FOREIGN KEY(id_tip_camera) REFERENCES tip_camera_OLAP(id_tip_camera);

ALTER TABLE rezervare_camera_OLAP

ADD FOREIGN KEY(id_tip_client) REFERENCES tip_client_OLAP(id_tip_client);

Script Output x



Task completed in 0,165 seconds

Table REZERVARE_CAMERA_OLAP altered.

Table REZERVARE_CAMERA_OLAP altered.

Table REZERVARE_CAMERA_OLAP altered.

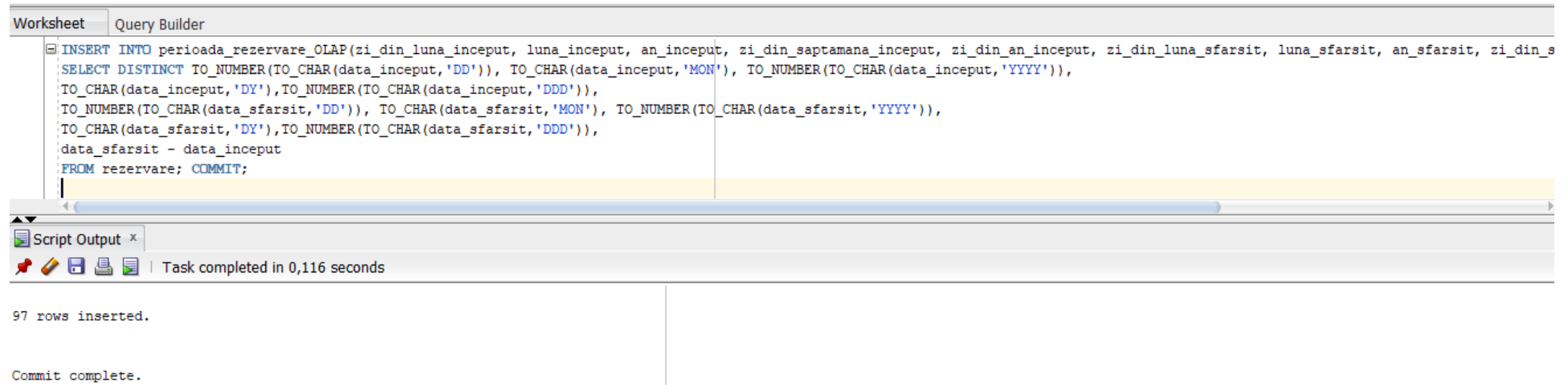
Table REZERVARE_CAMERA_OLAP altered.

Table REZERVARE_CAMERA_OLAP altered.

- 37 -

4. (0,5p) Popularea cu informații a bazei de date depozit folosind ca sursă datele din baza de date OLTP

```
INSERT INTO perioada_rezervare_OLAP(zi_din_luna_inceput, luna_inceput, an_inceput, zi_din_saptamana_inceput,
zi_din_an_inceput, zi_din_luna_sfarsit, luna_sfarsit, an_sfarsit, zi_din_saptamana_sfarsit, zi_din_an_sfarsit,
durata_in_zile)
SELECT DISTINCT TO_NUMBER(TO_CHAR(data_inceput,'DD')), TO_CHAR(data_inceput,'MON'),
TO_NUMBER(TO_CHAR(data_inceput,'YYYY')),
TO_CHAR(data_inceput,'DY'),TO_NUMBER(TO_CHAR(data_inceput,'DDD')),
TO_NUMBER(TO_CHAR(data_sfarsit,'DD')), TO_CHAR(data_sfarsit,'MON'), TO_NUMBER(TO_CHAR(data_sfarsit,'YYYY')),
TO_CHAR(data_sfarsit,'DY'),TO_NUMBER(TO_CHAR(data_sfarsit,'DDD')),
data_sfarsit - data_inceput
FROM rezervare; COMMIT;
```



```
INSERT INTO moment_efectuare_rezervare_OLAP(zi_din_luna, luna, an, zi_din_saptamana, zi_din_an)
SELECT DISTINCT TO_NUMBER(TO_CHAR(data_efectuarii,'DD')), TO_CHAR(data_efectuarii,'MON'),
TO_NUMBER(TO_CHAR(data_efectuarii,'YYYY')),TO_CHAR(data_efectuarii,'DY'),TO_NUMBER(TO_CHAR(data_efectuarii,'DD
D')) --,
--TO_NUMBER(TO_CHAR(data_efectuarii,'HH')) + CASE WHEN TO_NUMBER(TO_CHAR(data_efectuarii,'MI')) > 29 THEN 1
ELSE 0 END AS ora_aprox
FROM rezervare; COMMIT;
```

Worksheet Query Builder

```
INSERT INTO moment_efectuare_rezervare_OLAP(zi_din_luna, luna, an, zi_din_saptamana, zi_din_an)
SELECT DISTINCT TO_NUMBER(TO_CHAR(data_efectuarii,'DD')), TO_CHAR(data_efectuarii,'MON'),
TO_NUMBER(TO_CHAR(data_efectuarii,'YYYY')),TO_CHAR(data_efectuarii,'DY'),TO_NUMBER(TO_CHAR(data_efectuarii,'DDD'))--,
--TO_NUMBER(TO_CHAR(data_efectuarii,'HH')) + CASE WHEN TO_NUMBER(TO_CHAR(data_efectuarii,'MI')) > 29 THEN 1 ELSE 0 END AS ora_aprox
FROM rezervare; COMMIT;
```

Script Output x

Task completed in 0,095 seconds

84 rows inserted.

Commit complete.

```
INSERT INTO hotel_OLAP(id_hotel, nume, regiune, judet, localitate, pozitie, nr_stele, are_mic_dejun_inclus)
SELECT id_hotel, nume, regiune, judet, localitate, pozitie, nr_stele, are_mic_dejun_inclus
FROM hotel JOIN zona
USING(id_zona); COMMIT;
```

Worksheet Query Builder

```
INSERT INTO hotel_OLAP(id_hotel, nume, regiune, judet, localitate, pozitie, nr_stele, are_mic_dejun_inclus)
SELECT id_hotel, nume, regiune, judet, localitate, pozitie, nr_stele, are_mic_dejun_inclus
FROM hotel JOIN zona
USING(id_zona); COMMIT;
```

Script Output x

Task completed in 0,691 seconds

100 rows inserted.


Commit complete.

```

INSERT INTO tip_client_OLAP(varsta,gen,stare_civila)
SELECT DISTINCT FLOOR(MONTHS_BETWEEN(SYSDATE,data_nasterii)/12) AS varsta, gen, stare_civila
FROM utilizator; COMMIT;

```

Worksheet	Query Builder
	<pre> INSERT INTO tip_client_OLAP(varsta,gen,stare_civila) SELECT DISTINCT FLOOR(MONTHS_BETWEEN(SYSDATE,data_nasterii)/12) AS varsta, gen, stare_civila FROM utilizator; COMMIT; </pre>


Script Output x
 Task completed in 0,089 seconds
<pre> 84 rows inserted. Commit complete. </pre>

```

INSERT INTO tip_camera_OLAP(nr_paturi_duble, nr_paturi_simple,are_terasa,are_televizor)
SELECT DISTINCT nr_paturi_duble, nr_paturi_simple,are_terasa,are_televizor
FROM camera; COMMIT;

```

Worksheet	Query Builder
	<pre> INSERT INTO tip_camera_OLAP(nr_paturi_duble, nr_paturi_simple,are_terasa,are_televizor) SELECT DISTINCT nr_paturi_duble, nr_paturi_simple,are_terasa,are_televizor FROM camera; COMMIT; </pre>

Script Output x
 Task completed in 0,07 seconds
<pre> 36 rows inserted. Commit complete. </pre>

---functie ce calculeaza varsta clientului---

```
CREATE OR REPLACE FUNCTION calculeaza_varsta(data_nasterii DATE)
RETURN NUMBER IS
BEGIN
    RETURN FLOOR(MONTHS_BETWEEN(SYSDATE,data_nasterii)/12);
END calculeaza_varsta;
```



---functie ce cauta id-ul perioadei de rezervare---

```
CREATE OR REPLACE FUNCTION gaseste_id_perioada_OLAP(p_zi_din_an_inceput
perioada_rezervare_OLAP.zi_din_an_inceput%TYPE, p_an_inceput perioada_rezervare_OLAP.an_inceput%TYPE,
p_zi_din_an_sfarsit
perioada_rezervare_OLAP.zi_din_an_sfarsit%TYPE, p_an_sfarsit perioada_rezervare_OLAP.an_sfarsit%TYPE)
RETURN perioada_rezervare_OLAP.id_perioada%TYPE IS
    v_id_gasit perioada_rezervare_OLAP.id_perioada%TYPE;
BEGIN
    SELECT id_perioada
    INTO v_id_gasit
    FROM perioada_rezervare_OLAP
    WHERE zi_din_an_inceput=p_zi_din_an_inceput
    AND an_inceput=p_an_inceput
    AND zi_din_an_sfarsit=p_zi_din_an_sfarsit
    AND an_sfarsit= p_an_sfarsit;
    RETURN v_id_gasit;
END gaseste_id_perioada_OLAP;
```

Worksheet Query Builder

```

CREATE OR REPLACE FUNCTION gaseste_id_perioada_OLAP(p_zi_din_an_inceput perioada_rezervare_OLAP.zi_din_an_inceput%TYPE, p_an_inceput perioada_rezervare_OLAP.an_inceput%TYPE,
p_zi_din_an_sfarsit perioada_rezervare_OLAP.zi_din_an_sfarsit%TYPE, p_an_sfarsit perioada_rezervare_OLAP.an_sfarsit%TYPE)
RETURN perioada_rezervare_OLAP.id_perioada%TYPE IS
v_id_gasit perioada_rezervare_OLAP.id_perioada%TYPE;
BEGIN
SELECT id_perioada
INTO v_id_gasit
FROM perioada_rezervare_OLAP
WHERE zi_din_an_inceput=p_zi_din_an_inceput
AND an_inceput=p_an_inceput
AND zi_din_an_sfarsit=p_zi_din_an_sfarsit
AND an_sfarsit= p_an_sfarsit;
RETURN v_id_gasit;
END gaseste_id_perioada_OLAP;

```

Script Output x

Task completed in 0,096 seconds

Function GASESTE_ID_PERIOADA_OLAP compiled

---functie ce cauta id-ul momentului rezervarii---

```

CREATE OR REPLACE FUNCTION gaseste_id_moment_efectuare_OLAP(p_zi_din_an
moment_efectuare_rezervare_OLAP.zi_din_an%TYPE, p_an moment_efectuare_rezervare_OLAP.an%TYPE)
RETURN moment_efectuare_rezervare_OLAP.id_moment_efectuare%TYPE IS
v_id_gasit moment_efectuare_rezervare_OLAP.id_moment_efectuare%TYPE;
BEGIN
SELECT id_moment_efectuare
INTO v_id_gasit
FROM moment_efectuare_rezervare_OLAP
WHERE an=p_an
AND zi_din_an=p_zi_din_an;
RETURN v_id_gasit;
END gaseste_id_moment_efectuare_OLAP;

```

Worksheet Query Builder

```

CREATE OR REPLACE FUNCTION gaseste_id_moment_efectuare_OLAP(p_zi_din_an moment_efectuare_rezervare_OLAP.zi_din_an%TYPE, p_an moment_efectuare_rezervare_OLAP.an%TYPE)
RETURN moment_efectuare_rezervare_OLAP.id_moment_efectuare%TYPE IS
v_id_gasit moment_efectuare_rezervare_OLAP.id_moment_efectuare%TYPE;
BEGIN
SELECT id_moment_efectuare
INTO v_id_gasit
FROM moment_efectuare_rezervare_OLAP
WHERE an=p_an
AND zi_din_an=p_zi_din_an;
RETURN v_id_gasit;
END gaseste_id_moment_efectuare_OLAP;

```

Script Output x

Task completed in 0,078 seconds

Function GASESTE_ID_MOMENT_EFECTUARE_OLAP compiled

---functie ce cauta id-ul camerei---

```

CREATE OR REPLACE FUNCTION gaseste_id_tip_camera_OLAP(p_nr_paturi_duble tip_camera_OLAP.nr_paturi_duble%TYPE,
p_nr_paturi_simple tip_camera_OLAP.nr_paturi_duble%TYPE,
p_are_terasa tip_camera_OLAP.are_terasa%TYPE,
p_are televizor tip_camera_OLAP.are televizor%TYPE)
RETURN tip_camera_OLAP.id_tip_camera%TYPE IS
v_id_gasit tip_camera_OLAP.id_tip_camera%TYPE;
BEGIN
SELECT id_tip_camera
INTO v_id_gasit
FROM tip_camera_OLAP
WHERE nr_paturi_duble=p_nr_paturi_duble
AND nr_paturi_simple=p_nr_paturi_simple
AND are_terasa=p_are_terasa
AND are televizor=p_are televizor;
RETURN v_id_gasit;
END gaseste_id_tip_camera_OLAP;

```

Worksheet Query Builder

```

CREATE OR REPLACE FUNCTION gaseste_id_tip_camera_OLAP(p_nr_paturi_duble tip_camera_OLAP.nr_paturi_duble%TYPE, p_nr_paturi_simple tip_camera_OLAP.nr_paturi_duble%TYPE,
p_are_terasa tip_camera_OLAP.are_terasa%TYPE, p_are_televizor tip_camera_OLAP.are_televizor%TYPE)
RETURN tip_camera_OLAP.id_tip_camera%TYPE IS
v_id_gasit tip_camera_OLAP.id_tip_camera%TYPE;
BEGIN
SELECT id_tip_camera
INTO v_id_gasit
FROM tip_camera_OLAP
WHERE nr_paturi_duble=p_nr_paturi_duble
AND nr_paturi_simple=p_nr_paturi_simple
AND are_terasa=p_are_terasa
AND are_televizor=p_are_televizor;
RETURN v_id_gasit;
END gaseste_id_tip_camera_OLAP;

```

Script Output x

Task completed in 0,114 seconds

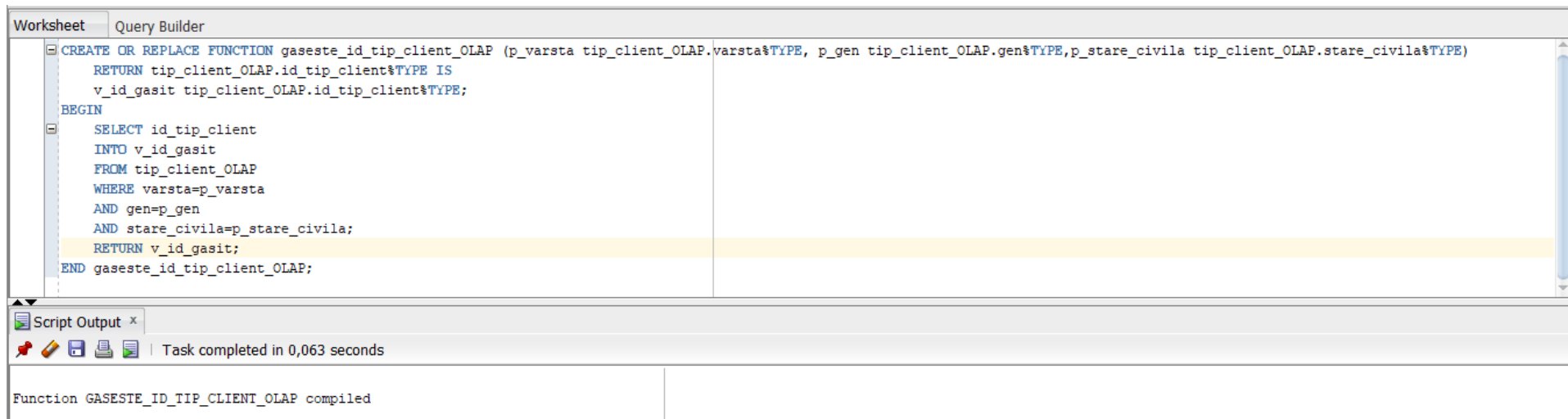
Function GASESTE_ID_TIP_CAMERA_OLAP compiled

---functie ce cauta id-ul clientului---

```

CREATE OR REPLACE FUNCTION gaseste_id_tip_client_OLAP (p_varsta tip_client_OLAP.varsta%TYPE, p_gen
tip_client_OLAP.gen%TYPE, p_stare_civila tip_client_OLAP.stare_civila%TYPE)
RETURN tip_client_OLAP.id_tip_client%TYPE IS
v_id_gasit tip_client_OLAP.id_tip_client%TYPE;
BEGIN
SELECT id_tip_client
INTO v_id_gasit
FROM tip_client_OLAP
WHERE varsta=p_varsta
AND gen=p_gen
AND stare_civila=p_stare_civila;
RETURN v_id_gasit;
END gaseste_id_tip_client_OLAP;

```



```

INSERT INTO
rezervare_camera_OLAP(id_rezervare,id_hotel,id_perioada,id_moment_efectuare,id_tip_camera,id_tip_client,pret)
SELECT DISTINCT id_rezervare,id_hotel,

gaseste_id_perioada_OLAP(TO_NUMBER(TO_CHAR(data_inceput,'DDD')),TO_NUMBER(TO_CHAR(data_inceput,'YYYY')),

TO_NUMBER(TO_CHAR(data_sfarsit,'DDD')),TO_NUMBER(TO_CHAR(data_sfarsit,'YYYY'))) AS id_perioada,

gaseste_id_moment_efectuare_OLAP(TO_NUMBER(TO_CHAR(data_efectuarii,'DDD')),TO_NUMBER(TO_CHAR(data_efectuarii,'
YYYY'))) AS id_moment_efectuare,

gaseste_id_tip_camera_OLAP(camera.nr_paturi_duble,camera.nr_paturi_simple,camera.are_terasa,camera.are_televiz
or) AS id_tip_camera,

gaseste_id_tip_client_OLAP(calculeaza_varsta(utilizator.data_nasterii),utilizator.gen,utilizator.stare_civila)
AS id_tip_client,

camera.pret_per_noapte * (rezervare.data_sfarsit - rezervare.data_inceput) AS pret
FROM utilizator JOIN rezervare
ON utilizator.id_utilizator=rezervare.id_client
JOIN rezervare_camera
USING(id_rezervare)
JOIN camera
USING(id_camera)
JOIN hotel
USING(id_hotel); COMMIT;

```

```
INSERT INTO rezervare_camera_OLAP(id_rezervare,id_hotel,id_perioada,id_moment_efectuare,id_tip_camera,id_tip_client,pret)
SELECT DISTINCT id_rezervare,id_hotel,
    gaseste_id_perioada_OLAP(TO_NUMBER(TO_CHAR(data_inceput,'DDD')),TO_NUMBER(TO_CHAR(data_inceput,'YYYY')),
        TO_NUMBER(TO_CHAR(data_sfarsit,'DDD')),TO_NUMBER(TO_CHAR(data_sfarsit,'YYYY'))) AS id_perioada,
    gaseste_id_moment_efectuare_OLAP(TO_NUMBER(TO_CHAR(data_efectuarii,'DDD')),TO_NUMBER(TO_CHAR(data_efectuarii,'YYYY'))) AS id_moment_efectuare,
    gaseste_id_tip_camera_OLAP(camera.nr_paturi_duble,camera.nr_paturi_simple,camera.are_terasa,camera.are_televizor) AS id_tip_camera,
    gaseste_id_tip_client_OLAP(calculeaza_varsta(utilizator.data_nasterii),utilizator.gen,utilizator.stare_civila) AS id_tip_client,
    camera.pret_per_noapte * (rezervare.data_sfarsit - rezervare.data_inceput) AS pret
FROM utilizator JOIN rezervare
ON utilizator.id_utilizator=rezervare.id_client
JOIN rezervare_camera
USING(id_rezervare)
JOIN camera
USING(id_camera)
JOIN hotel
USING(id_hotel); COMMIT;
```

Script Output x

Task completed in 0,341 seconds

149 rows inserted.

Commit complete.

5. (0,5p) Definirea constrângerilor

Constrângerile au fost definite încă de la momentul creării tabelului bazei de date depozit.

Exemple:

```
luna_inceput CHAR(3) CONSTRAINT luna_inceput_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
```

```
nr_paturi_duble NUMBER(1) CONSTRAINT nr_paturi_duble_nn_OLAP NOT NULL RELY DISABLE  
NOVALIDATE,
```

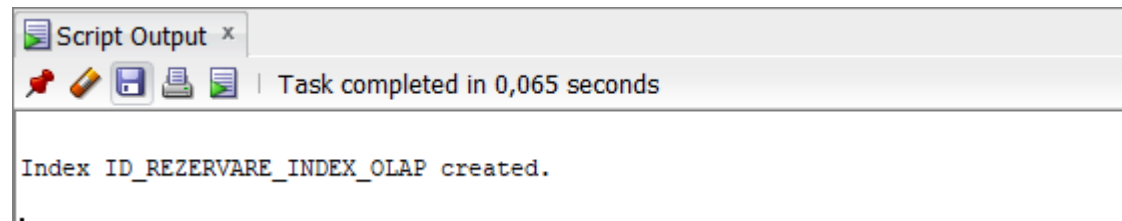
```
nr_stele NUMBER(1) CONSTRAINT nr_stele_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
```

```
varsta NUMBER(3) CONSTRAINT varsta_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
```

```
zi_din_an NUMBER(3,0) CONSTRAINT zi_din_an_nn_OLAP NOT NULL RELY DISABLE NOVALIDATE,
```

6. (1p) Definirea indecșilor și a cererilor SQL însoțite de planul de execuție al acestora (din care să reiasă ca optimizorul utilizează eficient indecșii definiți)

```
CREATE INDEX id_rezervare_index_OLAP  
ON rezervare_camera_OLAP(id_rezervare);
```



```
CREATE BITMAP INDEX luna_an_efectuare_bjindex_OLAP  
ON rezervare_camera_OLAP(moment_efectuare_OLAP.luna,moment_efectuare_OLAP.an)  
FROM moment_efectuare_OLAP,rezervare_camera_OLAP  
WHERE rezervare_camera_OLAP.id_moment_efectuare= moment_efectuare_OLAP.id_moment_efectuare;
```

```
CREATE BITMAP INDEX localitate_pozitie_bjindex_OLAP  
ON rezervare_camera_OLAP(hotel.localitate,hotel.pozitie)  
FROM hotel_OLAP, rezervare_camera_OLAP  
WHERE rezervare_camera_OLAP.id_hotel = hotel_OLAP.id_hotel;
```

```
CREATE BITMAP INDEX nr_paturi_duble_nr_paturi_simple_bjindex_OLAP  
ON rezervare_camera_OLAP(tip_camera_OLAP.nr_paturi_duble,tip_camera_OLAP.nr_paturi_simple)  
FROM tip_camera_OLAP, rezervare_camera_OLAP  
WHERE tip_camera_OLAP.id_tip_camera= rezervare_camera_OLAP.id_tip_camera;
```


Worksheet Query Builder

```
ON rezervare_camera_OLAP(id rezervare);

CREATE BITMAP INDEX luna_an_efectuare_bjindex_OLAP
ON rezervare_camera_OLAP(moment_efectuare_rezervare_OLAP.luna,moment_efectuare_rezervare_OLAP.an)
FROM moment_efectuare_rezervare_OLAP, rezervare_camera_OLAP
WHERE rezervare_camera_OLAP.id_moment_efectuare= moment_efectuare_rezervare_OLAP.id_moment_efectuare;

CREATE BITMAP INDEX localitate_pozitie_bjindex_OLAP
ON rezervare_camera_OLAP(hotel_OLAP.localitate,hotel_OLAP.pozitie)
FROM hotel_OLAP, rezervare_camera_OLAP
WHERE rezervare_camera_OLAP.id_hotel = hotel_OLAP.id_hotel;

CREATE BITMAP INDEX nr_paturi_duble_nr_paturi_simple_bjindex_OLAP
ON rezervare_camera_OLAP(tip_camera_OLAP.nr_paturi_duble,tip_camera_OLAP.nr_paturi_simple)
FROM tip_camera_OLAP, rezervare_camera_OLAP
WHERE tip_camera_OLAP.id_tip_camera= rezervare_camera_OLAP.id_tip_camera;
```

Script Output x

Task completed in 0.096 seconds

```
INDEX LOCALITATE_POZITIE_BJINDEX_OLAP created.

INDEX LUNA_AN_EFECTUARE_BJINDEX_OLAP created.

INDEX NR_PATURI_DUBLE_NR_PATURI_SIMPLE_BJINDEX_OLAP created.
```

Artistică 3D/Modelare

7. (1p) Definirea obiectelor de tip dimensiune, validarea acestora (din care să reiasă că datele respectă constrângerile impuse prin aceste tipuri de obiecte)

```
CREATE DIMENSION perioada_rezervare_dim_OLAP
LEVEL zi_din_an_inceput IS perioada_rezervare_OLAP.zi_din_an_inceput
LEVEL zi_din_an_sfarsit IS perioada_rezervare_OLAP.zi_din_an_sfarsit
ATTRIBUTE zi_din_an_inceput DETERMINES ( perioada_rezervare_OLAP.zi_din_luna_inceput,
perioada_rezervare_OLAP.luna_inceput, perioada_rezervare_OLAP.zi_din_saptamana_inceput)
ATTRIBUTE zi_din_an_sfarsit DETERMINES ( perioada_rezervare_OLAP.zi_din_luna_sfarsit,
perioada_rezervare_OLAP.luna_sfarsit, perioada_rezervare_OLAP.zi_din_saptamana_sfarsit);
```

The screenshot shows the 'Query Builder' window with the SQL script from the previous block. Below the script editor, the 'Script Output' pane displays the message 'Task completed in 0,202 seconds' and 'Dimension PERIODADA_REZERVARE_DIM_OLAP created.'

```
CREATE DIMENSION moment_efectuare_rezervare_dim_OLAP
LEVEL zi_din_an IS moment_efectuare_rezervare_OLAP.zi_din_an
ATTRIBUTE zi_din_an DETERMINES ( moment_efectuare_rezervare_OLAP.zi_din_luna,
moment_efectuare_rezervare_OLAP.luna, moment_efectuare_rezervare_OLAP.zi_din_saptamana);
```

The screenshot shows the 'Query Builder' window with the SQL script from the previous block. Below the script editor, the 'Script Output' pane displays the message 'Task completed in 0,081 seconds' and 'Dimension MOMENT_EFECTUARE_REZERVARE_DIM_OLAP created.'

--Ierarhia de mai jos este problematica deoarece exista posibilitatea ca numele a doua localitati din judete diferite sa coincida

```
CREATE DIMENSION hotel_dim_OLAP
LEVEL regiune IS hotel_OLAP.regiune
LEVEL judet IS hotel_OLAP.judet
LEVEL localitate IS hotel_OLAP.localitate
HIERARCHY judet_localitate (
    localitate CHILD OF
    judet CHILD OF
    regiune);
```

The screenshot shows the SQL Developer interface. The 'Worksheet' tab is active, displaying the following SQL script:

```
CREATE DIMENSION hotel_dim_OLAP
LEVEL regiune IS hotel_OLAP.regiune
LEVEL judet IS hotel_OLAP.judet
LEVEL localitate IS hotel_OLAP.localitate
HIERARCHY judet_localitate (
    localitate CHILD OF
    judet CHILD OF
    regiune
);
```

The 'Script Output' window is open, showing the message: 'Task completed in 0,142 seconds' and 'Dimension HOTEL_DIM_OLAP created.'

```
CREATE DIMENSION id_rezervare_dim_OLAP
LEVEL id_rezervare IS rezervare_camera_OLAP.id_rezervare
ATTRIBUTE id_rezervare DETERMINES (rezervare_camera_OLAP.id_perioada_rezervare,
rezervare_camera_OLAP.id_moment_efectuare_rezervare, rezervare_camera_OLAP.id_hotel,
rezervare_camera_OLAP.id_tip_client);
```

The screenshot shows the SQL Developer interface. The 'Worksheet' tab is active, displaying the following SQL script:

```
CREATE DIMENSION id_rezervare_dim_OLAP
LEVEL id_rezervare IS rezervare_camera_OLAP.id_rezervare
ATTRIBUTE id_rezervare DETERMINES (rezervare_camera_OLAP.id_perioada, rezervare_camera_OLAP.id_moment_efectuare, rezervare_camera_OLAP.id_hotel, rezervare_camera_OLAP.id_tip_client);
```

The 'Script Output' window is open, showing the message: 'Task completed in 0,065 seconds' and 'Dimension ID_REZERVARE_DIM_OLAP created.'

8. (1p) Definirea partițiilor; definirea cererilor SQL însoțite de planul de execuție al acestora din care să reiasă ca optimizorul utilizează eficient partițiile.

```
-pentru hotel
PARTITION BY LIST (nr_stele)
(PARTITION o_stea VALUES (1),
PARTITION doua_stele VALUES (2),
PARTITION trei_stele VALUES (3),
PARTITION patru_stele VALUES (4),
PARTITION cinci_stele VALUES (5));
```

```
-pentru rezervare_camera
PARTITION BY RANGE(pret)
(PARTITION pret_sub_1000 VALUES LESS THAN (1000),
PARTITION pret_sub_2500 VALUES LESS THAN (2500),
PARTITION pret_sub_5000 VALUES LESS THAN (5000),
PARTITION pret_peste_5000 VALUES LESS THAN (MAXVALUE));
```

9. (2p) Optimizarea cererii SQL propusă în etapa de analiză

a. (1p) planul de execuție ales de optimizorul bazat pe cost (explicație etape parcurse)

Se consideră cererea SQL ce afișează numele hotelurilor ce au avut suma rezervărilor cuprinsă între

```
SELECT nume, SUM(pret) AS suma
FROM hotel_OLAP, rezervare_camera_OLAP
where rezervare_camera_olap.id_hotel = hotel_olap.id_hotel
GROUP BY nume
HAVING SUM(pret) between 1000 and 2000;
```

Worksheet Query Builder

```

SELECT nume, SUM(pret) AS suma
FROM hotel_OLAP, rezervare_camera_OLAP
where rezervare_camera_olap.id_hotel = hotel_olap.id_hotel
GROUP BY nume
HAVING SUM(pret) between 1000 and 2000;

```

Script Output x Query Result x

SQL | All Rows Fetched: 9 in 0,032 seconds

	NUME	SUMA
1	Hohe Rinne Paltinis Hotel Spa	1720
2	Pensiunea Ioana	1410
3	Opera Plaza Hotel	1440
4	Ramada by Wyndham Oradea	1520
5	UpperHouse	1600
6	K+K Hotel Elisabeta	1300
7	Casa Vacanza Brasov	1000
8	Ivana Apart Hotel	1020
9	Hotel Central Park	1360

b. (1p) sugestii de optimizare a cererii, specificând planul de execuție obținut

Se poate crea o vizualizare materializată astfel:

```

CREATE MATERIALIZED VIEW pret_rezervare_camera_mw
BUILD IMMEDIATE
REFRESH FORCE
ENABLE QUERY REWRITE
AS
SELECT nume,
       SUM(pret) AS suma
FROM hotel_OLAP, rezervare_camera_OLAP
WHERE rezervare_camera_olap.id_hotel = hotel_olap.id_hotel
GROUP BY nume
HAVING SUM(pret) BETWEEN 10 AND 4000;

```

Worksheet Query Builder

```

CREATE MATERIALIZED VIEW pret_rezervare_camera_mw
BUILD IMMEDIATE
REFRESH FORCE
ENABLE QUERY REWRITE
AS
SELECT nume,
       SUM(pret) AS suma
FROM hotel_OLAP, rezervare_camera_OLAP
WHERE rezervare_camera_olap.id_hotel = hotel_olap.id_hotel
GROUP BY nume
HAVING SUM(pret) BETWEEN 10 AND 4000;

```

Script Output x

Task completed in 0,138 seconds

Materialized view PRET_REZERVARE_CAMERA_MW created.

Iar cererea se poate rescrie astfel:

```

SELECT nume, suma
FROM pret_rezervare_camera_mw
WHERE suma>100 AND suma<2000;

```

Worksheet Query Builder

```

SELECT nume, suma
FROM pret_rezervare_camera_mw
WHERE suma>100 AND suma<2000;

```

Script Output x Query Result x

SQL | All Rows Fetched: 17 in 0,012 seconds

NUME	SUMA
1 Pensiunea Christiana	570
2 Hohe Rinne Paltinis Hotel Spa	1720
3 Pensiunea Ioana	1410
4 Hotel Casa Wagner	990
5 Opera Plaza Hotel	1440
6 Hotel Arnia	210
7 Ramada by Wyndham Oradea	1520
8 UpperHouse	1600
9 Casa Rozelor boutique hotel	720
10 K+K Hotel Elisabeta	1300
11 Platinia Hotel	770
12 Casa Vacanza Brasov	1000
13 Golden Tulip Ana Dome	190
14 Ivana Apart Hotel	1020
15 Mandachi Hotel Spa	740
16 Hotel Central Park	1360
17 Pension Bellagio	450

10. (2p) Crearea rapoartelor cu complexitate diferită (la acest nivel vor fi scripturi SQL, fără reprezentare grafică)

```
SELECT moment_efectuare_rezervare_olap.luna lunaefectuare, COUNT(tip_camera_olap.id_tip_camera) numar,  
GROUPING_ID (luna) grouping_id  
FROM rezervare_camera_olap, moment_efectuare_rezervare_olap, tip_camera_olap  
WHERE rezervare_camera_olap.id_tip_camera= tip_camera_olap.id_tip_camera  
AND rezervare_camera_olap.id_moment_efectuare= moment_efectuare_rezervare_olap.id_moment_efectuare  
GROUP BY ROLLUP ( moment_efectuare_rezervare_olap.luna );
```



The screenshot shows a database query tool interface. The top pane displays the SQL query, and the bottom pane shows the query results in a table format. The results table has three columns: LUNAEFFECTUARE, NUMAR, and GROUPING_ID. The data is grouped by month, with each month having a GROUPING_ID of 0.

LUNAEFFECTUARE	NUMAR	GROUPING_ID
1 APR	13	0
2 AUG	10	0
3 DEC	10	0
4 FEB	8	0
5 IAN	19	0
6 IUL	21	0
7 IUN	10	0
8 MAI	9	0
9 MAR	13	0
10 NOI	14	0
11 OCT	16	0
12 SEP	6	0

PROIECT DATA WAREHOUSE

~ Etapa implementare aplicație ~

1. Modul aplicație prin care se introduc și gestionează informații la nivelul bazei de date OLTP

Pentru implementarea bazei de date am folosit ca sursa baza de date Oracle, versiunea 21c. Baza de date am creat-o direct într-un container, folosind Docker. Pentru obținerea imaginii bazei de date de tip oracle, am folosit site-ul oficial Oracle, <https://container-registry.oracle.com/>.

Am decis să folosim docker pentru a eficientiza procesul de instalare și configurare a bazei de date, care este utilizată de pe mai multe dispozitive.

Pentru backend & frontend, am folosit Python, framework-ul Django, prin care sub forma unei aplicații web putem altera și observa datele. Pentru managerarea containerelor, am folosit Docker Desktop, unde putem observa containerele rulant. La momentul creării containerelor, sunt rulate scripturi și fișiere de configurare care creează și populează bazele de date, respectiv tabele asociate OLTP și OLAP.

Pentru crearea containerelor, am folosit docker compose, unde avem cele 2 servicii definite:

<input checked="" type="checkbox"/>		NAME	IMAGE
<input checked="" type="checkbox"/>	▼	dockerdwbi	-
<input checked="" type="checkbox"/>		django d584ce1a0c13	dockerdwbi-django:latest
<input checked="" type="checkbox"/>		oracle 3965475dbf00	dockerdwbi-oracle:latest

STATUS	PORT(S)	STARTED	ACTIONS
Running (2/2)			
Running	8000:8000	1 hour ago	⏏ ⋮
Running	1521:1521 5500:5500	2 hours ago	⏏ ⋮

<input checked="" type="checkbox"/>		NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
<input checked="" type="checkbox"/>	▼	dockerdwbi	-	Running (2/2)			⏏ ⋮
<input checked="" type="checkbox"/>		django d584ce1a0c13	dockerdwbi-django:latest	Running	8000:8000	1 hour ago	⏏ ⋮
<input checked="" type="checkbox"/>		oracle 3965475dbf00	dockerdwbi-oracle:latest	Running	1521:1521 5500:5500	2 hours ago	⏏ ⋮


```

🐛 docker-compose.yml
1  version: '3'
2
3  services:
4    oracle:
5      build:
6        context: .
7        dockerfile: docker/Dockerfile.oracle
8      ports:
9        - 1521:1521
10       - 5500:5500
11     volumes:
12       - "./oracle:/ORCL"
13     environment:
14       - DB_SID=ORCLCDB
15       - ORACLE_SID=ORCLCDB
16       - DB_PDB=orclpdb1
17       - SYS_PASSWORD=Oradoc_db1
18     container_name: oracle
19   django:
20     build:
21       context: .
22       dockerfile: docker/Dockerfile.django
23     ports:
24       - "8000:8000"
25     volumes:
26       - ./usr/src/app/
27       - /tmp:/tmp
28     hostname: django
29     environment:
30       - RUN_DOCKERIZED=1
31       - PYTHONDONTWRITEBYTECODE=1
32       - PYTHONUNBUFFERED=1
33     container_name: django
34     depends_on:
35       - oracle

```

In script-urile de initializare, pe langa anumite conditii de conectivitate, vizibilitate a bazei de date intr-un sistem containerizat, incepem prin a crea userii, si tabelele OLTP.

```

alter session set "_ORACLE_SCRIPT"=true;

show con_name;
alter session set container= orclpdb1;
show con_name;
-- ALTER PLUGGABLE DATABASE orclpdb1 open; ALREADY OPEN?
-- alter system disable restricted session;
alter system set local_listener = '(ADDRESS=(PROTOCOL=TCP)(HOST=0.0.0.0)(PORT=1521))' scope = both;
CREATE USER marius IDENTIFIED BY db_pass;
GRANT ALL PRIVILEGES TO marius;
CREATE USER dw_manager IDENTIFIED BY mng_pass;
GRANT CREATE SESSION TO dw_manager;
GRANT CREATE ANY TABLE TO dw_manager;
GRANT CREATE ANY INDEX TO dw_manager;
GRANT CREATE VIEW TO dw_manager;

GRANT CREATE TRIGGER TO dw_manager;
GRANT CREATE ANY SEQUENCE TO dw_manager;
GRANT SELECT ANY TABLE TO dw_manager;
GRANT INSERT ANY TABLE TO dw_manager;
GRANT DELETE ANY TABLE TO dw_manager;
GRANT UPDATE ANY TABLE TO dw_manager;
GRANT ALTER ANY TABLE TO dw_manager;
GRANT UNLIMITED TABLESPACE TO dw_manager;

```

Script-urile de initializare, configurare sunt copiate in foldere predefinite de Oracle in scopul rularii acestora la startup.

```
SELECT *  
FROM session_privs;
```

```
SET FEEDBACK 1  
SET NUMWIDTH 10  
SET LINESIZE 80  
SET TRIMSPool ON  
SET TAB OFF  
SET PAGESIZE 100  
SET ECHO OFF
```

```
CREATE TABLE dw_manager.utilizator  
( id_utilizator NUMBER GENERATED ALWAYS as IDENTITY(START WITH 1 INCREMENT BY 1),  
  nume_utilizator VARCHAR(30) CONSTRAINT nume_utilizator_nn NOT NULL,  
  hash_parola VARCHAR(25) CONSTRAINT hash_parola_utilizator_nn NOT NULL,  
  nume_complet VARCHAR(30) CONSTRAINT nume_complet_utilizator_nn NOT NULL,
```

```
> Dockerfile.oracle > ...  
FROM container-registry.oracle.com/database/enterprise:latest  
  
COPY /config/01_create_user.sql /opt/oracle/scripts/startup/  
COPY /config/02_create_db.sql /opt/oracle/scripts/startup/  
COPY /config/03_populate_data.sql /opt/oracle/scripts/startup/  
COPY /config/create_olap_tables.sql /opt/oracle/scripts/startup/  
COPY /config/dmensions.sql /opt/oracle/scripts/startup/  
COPY /config/olap_functions.sql /opt/oracle/scripts/startup/  
COPY /config/olap_data.sql /opt/oracle/scripts/startup/  
COPY /config/olap_indexes.sql /opt/oracle/scripts/startup/  
  
COPY /config/database.sh /opt/oracle/scripts/setup/  
  
# CMD ["rm /opt/oracle/product/21c/dbhome_1/network/admin/samples/listener.ora"]  
  
COPY /config/listener.ora /opt/oracle/product/21c/dbhome_1/network/admin/samples/  
COPY /config/tnsnames.ora /opt/oracle/product/21c/dbhome_1/network/admin/samples/  
  
# CMD ["rm /opt/oracle/product/21c/dbhomeXE/network/admin/tnsnames.ora"]
```

Se putea obtine acelasi rezultat folosind un volum, dar pentru vizibilitate, am copiat script-urile in folderele predefinite.

```

023-01-26 20:13:09 DONE: Executing user defined scripts
023-01-26 20:13:09
023-01-26 20:13:09 The Oracle base remains unchanged with value /opt/oracle
023-01-26 20:13:10 #####
023-01-26 20:13:10 DATABASE IS READY TO USE!
023-01-26 20:13:10 #####
023-01-26 20:13:10
023-01-26 20:13:10 Executing user defined scripts
023-01-26 20:13:10 /opt/oracle/runUserScripts.sh: running /opt/oracle/scripts/startup/01_create_user.sql
023-01-26 20:13:10
023-01-26 20:13:10 Session altered.
023-01-26 20:13:10
023-01-26 20:13:10
023-01-26 20:13:10 CON_NAME
023-01-26 20:13:10 -----
023-01-26 20:13:10 CDB$ROOT
023-01-26 20:13:10
023-01-26 20:13:10 Session altered.
023-01-26 20:13:10
023-01-26 20:13:10
023-01-26 20:13:10 CON_NAME
023-01-26 20:13:10 -----
023-01-26 20:13:10 ORCLPDB1
023-01-26 20:13:10
023-01-26 20:13:10 System altered.
023-01-26 20:13:10
023-01-26 20:13:10
023-01-26 20:13:10 User created.
023-01-26 20:13:10
023-01-26 20:13:10
023-01-26 20:13:10 Grant succeeded.
023-01-26 20:13:10
023-01-26 20:13:10
023-01-26 20:13:10 User created.
023-01-26 20:13:10
023-01-26 20:13:10
023-01-26 20:13:10 Grant succeeded.

```

Dupa crearea tabelelor, si a userilor asociati, se insereaza datele, tot din cadrul fisierilor care ruleaza la startup:

```

138 ALTER TABLE dw_manager.atribuie
139 ADD CONSTRAINT fk_atribuie_camera FOREIGN KEY (id_camera) REFERENCES dw_manager.camera(id_camera);
140
141
142 alter session set container= orclpdb1;
143
144 INSERT INTO dw_manager.zona (JUDET, LOCALITATE, POZITIE) VALUES ('Mures', 'Targu Mures', 'centrala');
145 INSERT INTO dw_manager.zona (JUDET, LOCALITATE, POZITIE) VALUES ('Brasov', 'Bran ', 'centrala');
146 INSERT INTO dw_manager.zona (JUDET, LOCALITATE, POZITIE) VALUES ('Brasov', 'Bran ', 'periferica');
147 INSERT INTO dw_manager.zona (JUDET, LOCALITATE, POZITIE) VALUES ('Bucharest', 'Bucharest', 'centrala');
148 INSERT INTO dw_manager.zona (JUDET, LOCALITATE, POZITIE) VALUES ('Prahova', 'Sinaia ', 'centrala');
149 INSERT INTO dw_manager.zona (JUDET, LOCALITATE, POZITIE) VALUES ('Bucharest', 'Bucharest ', 'periferica');

```

[illegible][illegible]

Tabele pentru baza de date OLAP se populeaza din cadrul urmatoarei script:

```
INSERT INTO perioada_rezervare_OLAP(zi_din_luna_inceput, luna_inceput, an_inceput, zi_din_saptamana_inceput,
zi_din_an_inceput, zi_din_luna_sfarsit, luna_sfarsit, an_sfarsit, zi_din_saptamana_sfarsit, zi_din_an_sfarsit,
durata_in_zile)

SELECT DISTINCT TO_NUMBER(TO_CHAR(data_inceput,'DD')), TO_CHAR(data_inceput,'MON'),
TO_NUMBER(TO_CHAR(data_inceput,'YYYY')),
TO_CHAR(data_inceput,'DY'),TO_NUMBER(TO_CHAR(data_inceput,'DDD')),
TO_NUMBER(TO_CHAR(data_sfarsit,'DD')), TO_CHAR(data_sfarsit,'MON'), TO_NUMBER(TO_CHAR(data_sfarsit,'YYYY')),
TO_CHAR(data_sfarsit,'DY'),TO_NUMBER(TO_CHAR(data_sfarsit,'DDD')),
data_sfarsit - data_inceput
FROM rezervare; COMMIT;

INSERT INTO moment_efectuare_rezervare_OLAP(zi_din_luna, luna, an, zi_din_saptamana, zi_din_an)

SELECT DISTINCT TO_NUMBER(TO_CHAR(data_efectuarii,'DD')), TO_CHAR(data_efectuarii,'MON'),
TO_NUMBER(TO_CHAR(data_efectuarii,'YYYY')),TO_CHAR(data_efectuarii,'DY'),TO_NUMBER(TO_CHAR(data_efectuarii,'DDD'))--,
--TO_NUMBER(TO_CHAR(data_efectuarii,'HH')) + CASE WHEN TO_NUMBER(TO_CHAR(data_efectuarii,'MI')) > 29 THEN 1 ELSE 0 END
AS ora_aprox
FROM rezervare; COMMIT;

INSERT INTO hotel_OLAP(id_hotel, nume, regiune, judet, localitate, pozitie, nr_stele, are_mic_dejun_inclus)

SELECT id_hotel, nume, regiune, judet, localitate, pozitie, nr_stele, are_mic_dejun_inclus
FROM hotel JOIN zona
USING(id_zona); COMMIT;
```

```

INSERT INTO tip_client_OLAP(varsta,gen,stare_civila)
SELECT DISTINCT FLOOR(MONTHS_BETWEEN(SYSDATE,data_nasterii)/12) AS varsta, gen, stare_civila
FROM utilizator; COMMIT;

INSERT INTO tip_camera_OLAP(nr_paturi_duble, nr_paturi_simple,are_terasa,are_televizor)
SELECT DISTINCT nr_paturi_duble, nr_paturi_simple,are_terasa,are_televizor
FROM camera; COMMIT;

INSERT INTO rezervare_camera_OLAP
(id_rezervare,id_hotel,id_perioada,id_moment_efectuare,id_tip_camera,id_tip_client,pret)
SELECT DISTINCT id_rezervare,id_hotel,
gaseste_id_perioada_OLAP(TO_NUMBER(TO_CHAR(data_inceput,'DDD')),
TO_NUMBER(TO_CHAR(data_inceput,'YYYY')),
TO_NUMBER(TO_CHAR(data_sfarsit,'DDD')),
TO_NUMBER(TO_CHAR(data_sfarsit,'YYYY')) AS id_perioada,
gaseste_id_moment_efectuare_OLAP(TO_NUMBER(TO_CHAR(data_efectuarii,'DDD')),
TO_NUMBER(TO_CHAR(data_efectuarii,'YYYY')) AS id_moment_efectuare,
gaseste_id_tip_camera_OLAP(camera.nr_paturi_duble,camera.nr_paturi_simple,
camera.are_terasa,camera.are_televizor) AS id_tip_camera,
gaseste_id_tip_client_OLAP(calculeaza_varsta(utilizator.data_nasterii),utilizator.gen,
utilizator.stare_civila) AS id_tip_client,
camera.pret_per_noapte * (rezervare.data_sfarsit - rezervare.data_inceput)
AS pret
FROM utilizator JOIN rezervare

```



```

ON utilizator.id_utilizator=rezervare.id_client
JOIN rezervare_camera
USING(id_rezervare)
JOIN camera
USING(id_camera)
JOIN hotel
USING(id_hotel); COMMIT;

```

Dupa rularea script-ului, observam in docker aceleasi log-uri:

2023-01-31 20:27:03	1 row created.	2023-01-31 20:27:03	1 row created.
2023-01-31 20:27:03		2023-01-31 20:27:03	
2023-01-31 20:27:03		2023-01-31 20:27:03	
2023-01-31 20:27:03	1 row created.	2023-01-31 20:27:03	1 row created.
2023-01-31 20:27:03		2023-01-31 20:27:03	
2023-01-31 20:27:03		2023-01-31 20:27:03	
2023-01-31 20:27:03	1 row created.	2023-01-31 20:27:03	1 row created.
2023-01-31 20:27:03		2023-01-31 20:27:03	
2023-01-31 20:27:03		2023-01-31 20:27:03	
2023-01-31 20:27:03	1 row created.	2023-01-31 20:27:03	1 row created.
2023-01-31 20:27:03		2023-01-31 20:27:03	
2023-01-31 20:27:03		2023-01-31 20:27:03	
2023-01-31 20:27:03	1 row created.	2023-01-31 20:27:03	1 row created.
2023-01-31 20:27:03		2023-01-31 20:27:03	
2023-01-31 20:27:03		2023-01-31 20:27:03	
2023-01-31 20:27:03	1 row created.	2023-01-31 20:27:03	1 row created.
2023-01-31 20:27:03		2023-01-31 20:27:03	
2023-01-31 20:27:03		2023-01-31 20:27:03	

2. Posibilitatea de vizualizare a datelor introduse/actualizate la nivelul bazei de date OLTP și posibilitatea de a verifica propagarea acestor operații asupra datelor din baza de date depozit

Aplicatie – Frontend,

(Back end) Datele de tip OLTP au fost proiectate in back end, sub forma unor modele, folosind ORM-ul django. Comanda *python manage.py inspectdb* genereaza aceste tabele, care trebuie putin customizate pentru a functiona in diferite scenarii.

```
from django.db import models

# Create your models here.

class Atribuie(models.Model):
    id_rezervare = models.OneToOneField('Rezervare', models.DO_NOTHING, db_column='id_rezervare', primary_key=True)
    id_camera = models.OneToOneField('Camera', models.DO_NOTHING, db_column='id_camera')

    class Meta:
        # managed = False
        db_table = 'atribuie'
        unique_together = (('id_rezervare', 'id_camera'),)
```

Aceste modele sunt vizibile in interfata web a aplicatiei django, printr-o pagina de management alocata administratorului bazei de date.

```

class Camera(models.Model):
    id_camera = models.FloatField(primary_key=True, blank=True)
    id_hotel = models.FloatField()
    nr_camera = models.FloatField(blank=True, null=True)
    nr_etaj = models.FloatField(blank=True, null=True)
    nr_paturi_duble = models.FloatField()
    nr_paturi_simple = models.FloatField()
    are_terasa = models.BooleanField()
    are_televizor = models.BooleanField()
    pret_per_noapte = models.FloatField()

    def __str__(self) -> str:
        return "Camera " + str(self.id_camera)

class Meta:
    # managed = False
    db_table = 'camera'

```

Dupa introducerea datelor de logare definite in fisierele de config ale oracle, admin-ul va putea vedea toate datele de tip OLTP din cadrul bazei de date .

```

34     def _do_insert(self, manager, using, fields, update_pk, raw):
35         fields = [
36             f for f in fields if f.attname not in ['id_camera']
37         ]
38         return super()._do_insert(manager, using, fields, update_pk, raw)
39
40     def _do_update(self, base_qs, using, pk_val, values, update_fields, forced_update):
41         values = [
42             value for value in values if value[0].attname not in ['id_camera']
43         ]

```

Django administration

Username:

Password:

Log in

Django administration

Site administration

AUTHENTICATION AND AUTHORIZATION






Groups	+ Add	Change
Users	+ Add	Change

EROOKING

Atribuíes	+ Add	Change
Cameras	+ Add	Change
Hotels	+ Add	Change
Rezervares	+ Add	Change
Utilizators	+ Add	Change
Zonas	+ Add	Change

Recent actions

My actions

-  Camera 659.0
Camera
-  Camera 659.0
Camera
-  Camera 659.0
Camera
-  Camera object (None)
Camera
-  Zona object (1231.0)
Zona

Change camera

Camera 1099

Id camera:

1099

Id hotel:

76

Nr camera:

527

Nr etaj:

5

Nr paturi duble:

0

Nr paturi simple:

2

☒ Are terasa☐ Are televizor

Pret per noapte:

180

[Save and add another](#)[Save and continue editing](#)[SAVE](#)[Delete](#)

3. *Rapoartele grafice asociate cererilor definite în etapele anterioare (punctul 10).*

Pe pagina principala, <http://localhost:8000/>, ne sunt prezentate graficele corespunzatoare script-urilor sql create în continuare.

Crearea rapoartelor cu complexitate diferită

Pentru aceste rapoarte au fost create grafice de tip chart in front-end aplicatiei Web.

1* - Cea mai scumpa camera per noapte din romania

```
select max(pret_per_noapte) from
(select hotel.ume,camera.pret_per_noapte from hotel
INNER JOIN camera on camera.id_hotel = hotel.id_hotel
order by hotel.ume)
```

2* Cea mai ieftina camera dintr-o zona centrala

```
select * from (select hotel.ume, MIN(pret_per_noapte) from camera
INNER JOIN hotel
on hotel.id_hotel=camera.id_hotel
INNER JOIN zona on zona.id_zona = hotel.id_zona
WHERE zona.pozitie = 'centrala'
GROUP BY hotel.ume)
where rownum = 1
```

3* Hotelul cu cele mai multe rezervari efectuate

```
select * from(  
SELECT MAX(rezervari) as mres, numes FROM (SELECT hotel.nume as  
numes,COUNT(atribuie.id_rezervare) as rezervari from atribuie  
inner join camera on camera.id_camera=atribuie.id_camera  
inner join hotel on hotel.id_hotel = camera.id_hotel  
GROUP BY hotel.nume)  
group by numes  
ORDER BY mres DESC)  
where rownum=1;
```

4* Cele mai scumpe 5 hoteluri

```
select DISTINCT hotel.nume,camera.pret_per_noapte from hotel  
INNER JOIN camera  
on camera.id_hotel = hotel.id_hotel  
order by camera.pret_per_noapte desc  
fetch first 5 rows only;
```

Welcome Page x hotels_mng x

Worksheet Query Builder

```

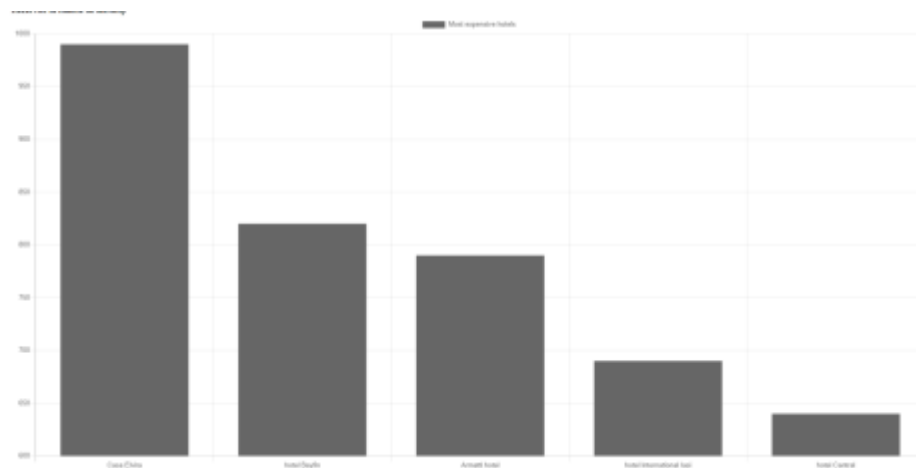
select DISTINCT hotel.num, camera.pret_per_noapte from hotel
INNER JOIN camera
on camera.id_hotel = hotel.id_hotel
order by camera.pret_per_noapte desc
fetch first 5 rows only;

```

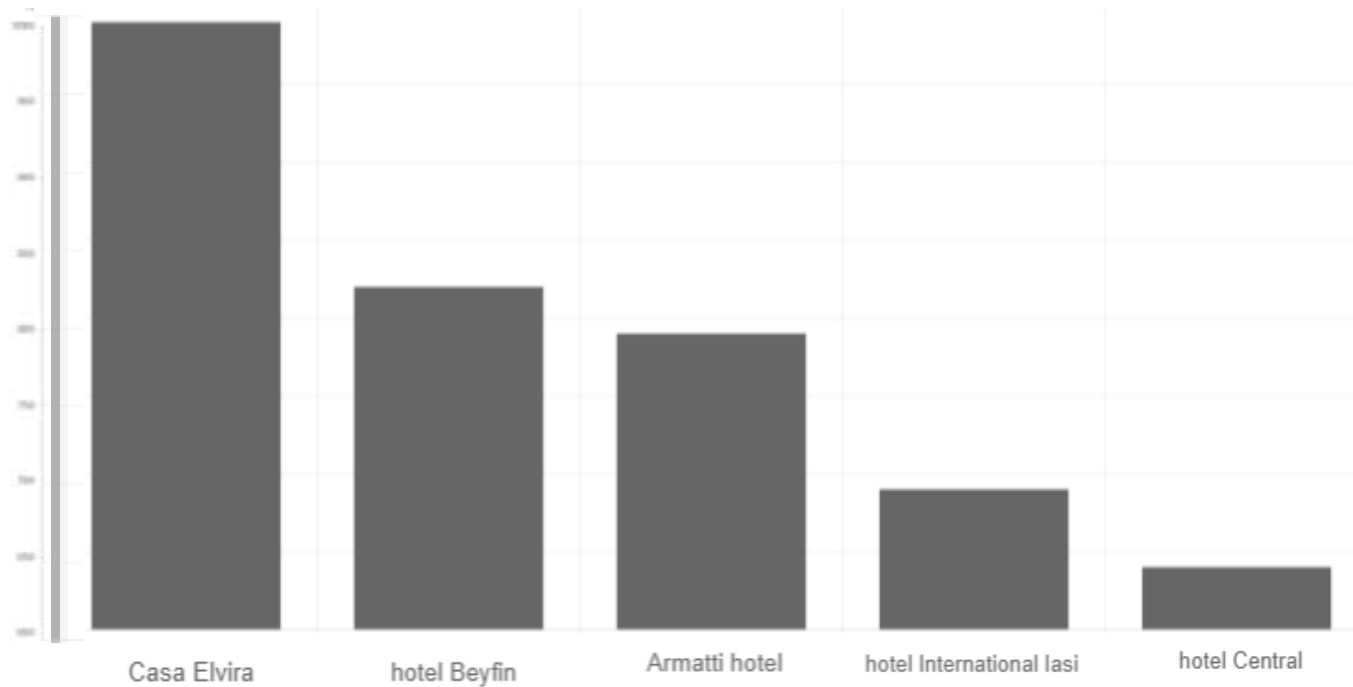
Query Result x

SQL | All Rows Fetched: 5 in 0.057 seconds

	NUME	PRET_PER_NOAPTE
1	Casa Elvira	990
2	hotel Beyfin	820
3	Armatti hotel	790
4	hotel International Iasi	690
5	hotel Central	640



Cele mai scumpe hoteluri 5 din Romania



5* Rezervari in functie de luna anului

```
SELECT COUNT(*) FROM REZERVARE
WHERE EXTRACT(MONTH FROM data_inceput) IN (1,4)
SELECT COUNT(*) FROM REZERVARE
WHERE EXTRACT(MONTH FROM data_inceput) IN (4, 7)
SELECT COUNT(*) FROM REZERVARE
WHERE EXTRACT(MONTH FROM data_inceput) IN (7, 10)
SELECT COUNT(*) FROM REZERVARE
WHERE EXTRACT(MONTH FROM data_inceput) IN (10, 1)
```

```

bar_data = []

with connection.cursor() as cursor:
    cursor.execute("SELECT COUNT(*) FROM REZERVARE WHERE EXTRACT(MONTH FROM data_inceput) IN (1,4)")
    first_3_months = cursor.fetchone()
    cursor.execute("SELECT COUNT(*) FROM REZERVARE WHERE EXTRACT(MONTH FROM data_inceput) IN (4,7)")
    first_6_months = cursor.fetchone()
    cursor.execute("SELECT COUNT(*) FROM REZERVARE WHERE EXTRACT(MONTH FROM data_inceput) IN (7,10)")
    first_9_months = cursor.fetchone()
    cursor.execute("SELECT COUNT(*) FROM REZERVARE WHERE EXTRACT(MONTH FROM data_inceput) IN (10,12)")
    first_12_months = cursor.fetchone()

```

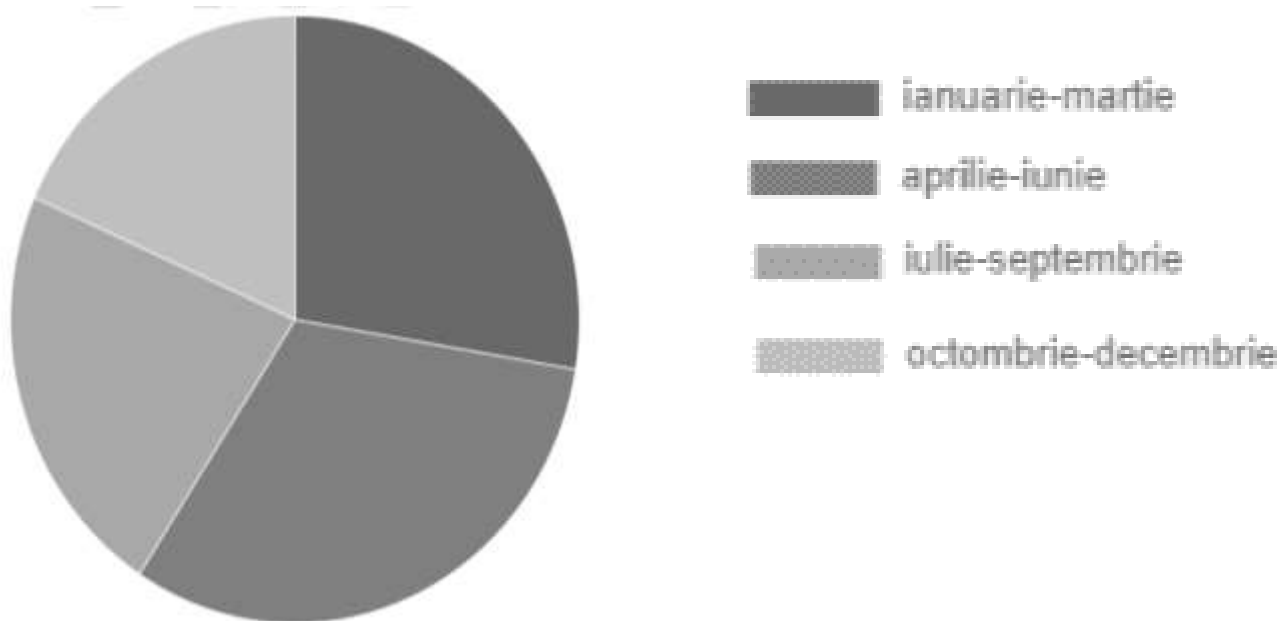
```

labels.extend(['ianuarie-martie', 'aprilie-iunie', 'iulie-septembrie', 'octombrie-decembrie'])
data.extend([first_3_months[0], first_6_months[0], first_9_months[0], first_12_months[0]])

print("LABELS:", labels)
print("DATA :", data)

# *** Most expensive hotels ***

```



Cereri in functie de perioada anului

```
with connection.cursor() as cursor:
    cursor.execute("select DISTINCT hotel.nume, camera.pret_per_noapte from hotel
        INNER JOIN camera on camera.id_hotel = hotel.id_hotel
        order by camera.pret_per_noapte desc fetch first 5 rows only;")
    most_expensive = cursor.fetchall()

    print("MOST EXPENSIVE: ", most_expensive)

    for hotel in most_expensive:
        bar_labels.append(hotel[0])
        bar_data.append(hotel[1])
```