TEMA: L1. Proiectarea studiului de caz

Tema: Platformă pentru analiza pieței imobiliare

Indicativ_echipa: SIA_10

Studenți: Diaconu Claudia, Achițeni Geanina-Ionela

Grupa 1, anul 2

STABILIRE surse de date externe

● DS_1: realtor-data.csv

Fișierul **realtor-data.csv** servește ca o bază pentru analiza pieței imobiliare, facilitând studiul prețurilor, tipurilor de proprietăți și tranzacțiilor imobiliare anterioare.

- Tip sursă de date
  - Tip model de date: relațional.
  - Tip format de acces: CSV.
- Descriere structuri de date
  - câmpuri/coloane/atribute etc.;
    - *brokered_by*: Agenția sau agentul imobiliar care a intermediat vânzarea.
    - *status:* Starea proprietății (de exemplu, "pentru vânzare").
    - *price*: Prețul de vânzare al proprietății.
    - *bed*: Numărul de dormitoare ale proprietății.
    - *bath*: Numărul de băi ale proprietății.
    - *acre_lot*: Dimensiunea terenului (în acri).
    - *street*: Strada pe care se află proprietatea.
    - *city*: Orașul în care se află proprietatea.
    - *state*: Statul în care se află proprietatea.
    - *zip_code*: Codul poștal al locației proprietății.
    - *house_size*: Dimensiunea casei în metri pătrați.
    - *prev_sold_date*: Data anterioară când proprietatea a fost vândută.

- legături între structurile din cadrul aceleași surse de date.
    - Proprietăți pot fi legate de un agent imobiliar(indicar prin brokered_by), iar prețurile și dimenisiunile sunt legate de locația propriu-zisă.
    - Prețurile sunt influențate de dimensiunile și starea proprietății.
    - Fiecare proprietate este localizată într-o anumită stradă, oraș, stat și cod poștal.

○ Implementare sursa de date externa

■ Script_Comenzi_DDL_DML (pentru format SQL/NoSQL)

În **oracle sql developer** am creat patru tabele: cities, owners, property_types, și real_estate.

```
-- 1. Orașe
CREATE TABLE cities (
  id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  city_name VARCHAR2(100) NOT NULL,
  state VARCHAR2(50) NOT NULL,
  zip_code VARCHAR2(10),
  UNIQUE (city_name, state)
);

-- 2. Proprietari
CREATE TABLE owners (
  id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  full_name VARCHAR2(100) NOT NULL,
  email VARCHAR2(100),
  phone VARCHAR2(20),
  created_at DATE DEFAULT SYSDATE
);
```

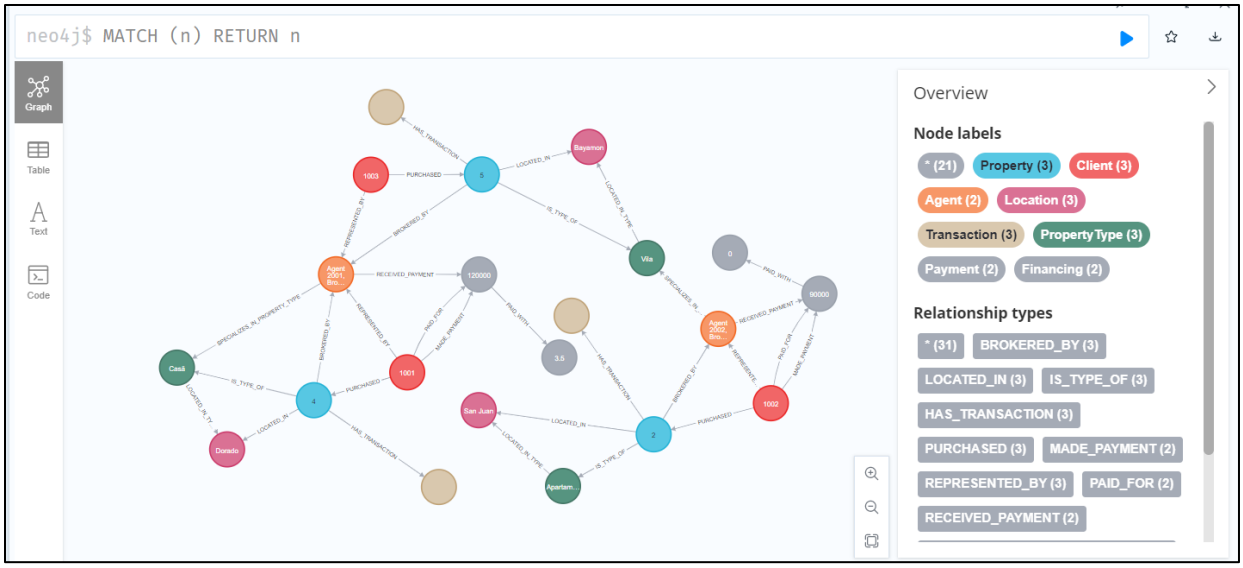
```sql
-- 3. Tipuri de proprietăți
CREATE TABLE property_types (
  id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  type_name VARCHAR2(50) NOT NULL CHECK (type_name IN ('House', 'Apartment', 'Condo', 'Villa')),
  description VARCHAR2(200)
);

-- 4. Proprietăți imobiliare
CREATE TABLE real_estate (
  id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  price NUMBER(10,2) NOT NULL CHECK (price > 0),
  beds NUMBER(2) CHECK (beds >= 0),
  baths NUMBER(2) CHECK (baths >= 0),
  house_size NUMBER(6) CHECK (house_size > 0),
  listed_date DATE DEFAULT SYSDATE,
  is_available CHAR(1) DEFAULT 'Y' CHECK (is_available IN ('Y', 'N')),
  city_id NUMBER NOT NULL,
  property_type_id NUMBER NOT NULL,
  owner_id NUMBER NOT NULL,
  FOREIGN KEY (city_id) REFERENCES cities(id),
  FOREIGN KEY (property_type_id) REFERENCES property_types(id),
  FOREIGN KEY (owner_id) REFERENCES owners(id)
);
```

Am facut inserturi pentru fiecare tabelă în parte.

```sql
INSERT INTO cities (city_name, state, zip_code) VALUES ('Austin', 'TX', '73301');
INSERT INTO cities (city_name, state, zip_code) VALUES ('Miami', 'FL', '33101');
INSERT INTO cities (city_name, state, zip_code) VALUES ('Chicago', 'IL', '60601');
INSERT INTO cities (city_name, state, zip_code) VALUES ('New York', 'NY', '10001');
INSERT INTO cities (city_name, state, zip_code) VALUES ('Los Angeles', 'CA', '90001');
INSERT INTO cities (city_name, state, zip_code) VALUES ('Houston', 'TX', '77001');
INSERT INTO cities (city_name, state, zip_code) VALUES ('Dallas', 'TX', '75201');
INSERT INTO cities (city_name, state, zip_code) VALUES ('Phoenix', 'AZ', '85001');
INSERT INTO cities (city_name, state, zip_code) VALUES ('Seattle', 'WA', '98101');
INSERT INTO cities (city_name, state, zip_code) VALUES ('Denver', 'CO', '80201');
INSERT INTO cities (city_name, state, zip_code) VALUES ('Atlanta', 'GA', '30301');
INSERT INTO cities (city_name, state, zip_code) VALUES ('San Diego', 'CA', '92101');
INSERT INTO cities (city_name, state, zip_code) VALUES ('Boston', 'MA', '02101');
INSERT INTO cities (city_name, state, zip_code) VALUES ('Orlando', 'FL', '32801');



INSERT INTO owners (full_name, email, phone) VALUES ('Owner 1', 'owner1@example.com', '0710000001');
INSERT INTO owners (full_name, email, phone) VALUES ('Owner 2', 'owner2@example.com', '0710000002');
INSERT INTO owners (full_name, email, phone) VALUES ('Owner 3', 'owner3@example.com', '0710000003');
INSERT INTO owners (full_name, email, phone) VALUES ('Owner 4', 'owner4@example.com', '0710000004');
INSERT INTO owners (full_name, email, phone) VALUES ('Owner 5', 'owner5@example.com', '0710000005');
INSERT INTO owners (full_name, email, phone) VALUES ('Owner 6', 'owner6@example.com', '0710000006');
INSERT INTO owners (full_name, email, phone) VALUES ('Owner 7', 'owner7@example.com', '0710000007');
INSERT INTO owners (full_name, email, phone) VALUES ('Owner 8', 'owner8@example.com', '0710000008');
INSERT INTO owners (full_name, email, phone) VALUES ('Owner 9', 'owner9@example.com', '0710000009');
INSERT INTO owners (full_name, email, phone) VALUES ('Owner 10', 'owner10@example.com', '0710000010');
```

```sql
INSERT INTO property_types (type_name, description) VALUES ('House', 'Description for House');
INSERT INTO property_types (type_name, description) VALUES ('Apartment', 'Description for Apartment');
INSERT INTO property_types (type_name, description) VALUES ('Condo', 'Description for Condo');
INSERT INTO property_types (type_name, description) VALUES ('Villa', 'Description for Villa');
INSERT INTO property_types (type_name, description) VALUES ('Studio', 'Description for Studio');
INSERT INTO property_types (type_name, description) VALUES ('Penthouse', 'Description for Penthouse');
INSERT INTO property_types (type_name, description) VALUES ('Mansion', 'Description for Mansion');
INSERT INTO property_types (type_name, description) VALUES ('Townhouse', 'Description for Townhouse');
INSERT INTO property_types (type_name, description) VALUES ('Duplex', 'Description for Duplex');
INSERT INTO property_types (type_name, description) VALUES ('Ranch', 'Description for Ranch');



INSERT INTO real_estate (price, beds, baths, house_size, listed_date, is_available, city_id, property_type_id, owner_id)
VALUES (350000, 4, 3, 2500, TO_DATE('2023-10-01', 'YYYY-MM-DD'), 'Y', 1, 1, 1);

INSERT INTO real_estate (price, beds, baths, house_size, listed_date, is_available, city_id, property_type_id, owner_id)
VALUES (400000, 5, 4, 3500, TO_DATE('2023-10-02', 'YYYY-MM-DD'), 'Y', 2, 2, 2);

INSERT INTO real_estate (price, beds, baths, house_size, listed_date, is_available, city_id, property_type_id, owner_id)
VALUES (250000, 3, 2, 1800, TO_DATE('2023-09-15', 'YYYY-MM-DD'), 'N', 3, 3, 3);

INSERT INTO real_estate (price, beds, baths, house_size, listed_date, is_available, city_id, property_type_id, owner_id)
VALUES (500000, 6, 5, 4500, TO_DATE('2023-11-05', 'YYYY-MM-DD'), 'Y', 4, 4, 4);

INSERT INTO real_estate (price, beds, baths, house_size, listed_date, is_available, city_id, property_type_id, owner_id)
VALUES (600000, 7, 6, 6000, TO_DATE('2023-12-01', 'YYYY-MM-DD'), 'N', 5, 5, 5);

INSERT INTO real_estate (price, beds, baths, house_size, listed_date, is_available, city_id, property_type_id, owner_id)
VALUES (450000, 4, 3, 3500, TO_DATE('2023-10-10', 'YYYY-MM-DD'), 'Y', 6, 1, 6);
```

Fișierul **realtor-data.csv** l-am folosit și pentru crearea unui graf cu noduri și relații între ele în **Neo4j**.



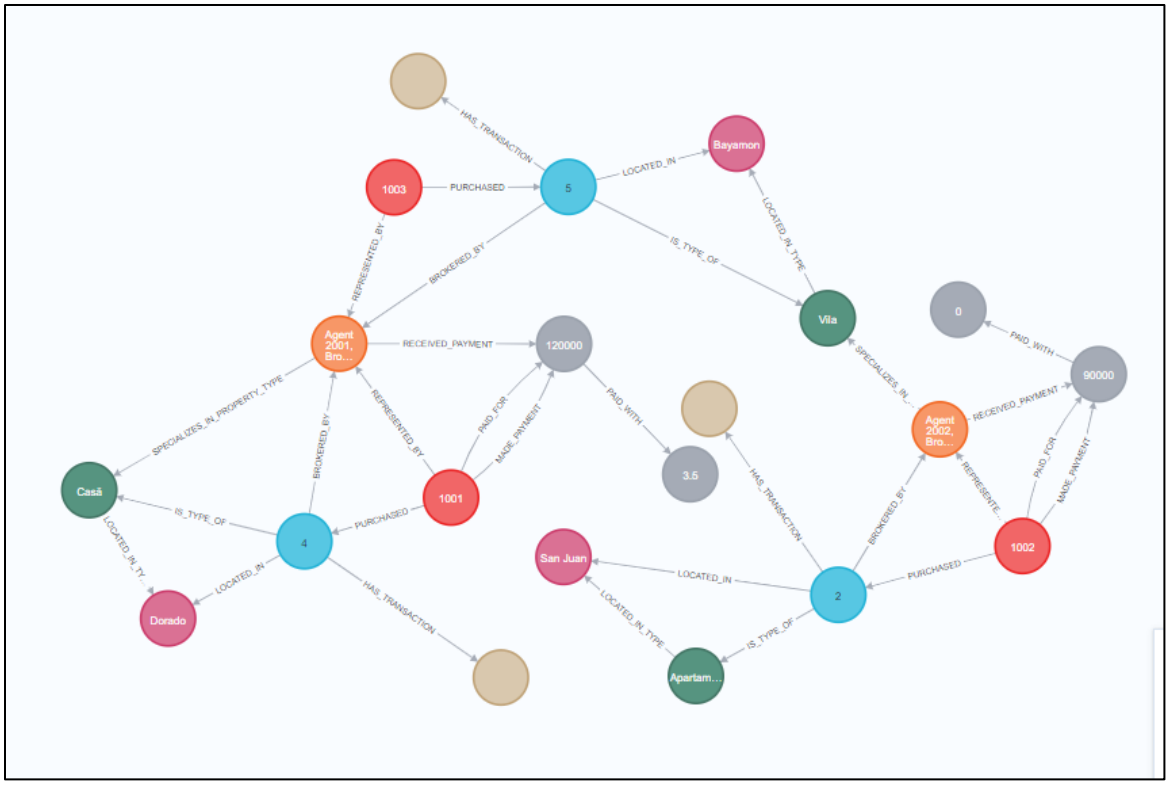
- **DS_2: data.csv**

Fișierul **data.csv** conține informații despre facturile emise, detalii ale produselor vândute și clienților care au efectuat achiziții. Acesta poate fi utilizat pentru analiza vânzărilor, comportamentului clienților și performanței produselor.

- Tip sursă de date
  - Tip model de date: graf.
  - Tip format de access: CSV.
- Descriere structuri de date
  - câmpuri/coloane/atribute etc.;
    - *InvoiceNo*: Numărul facturii, identificator unic pentru fiecare tranzacție.
    - *StockCode*: Codul produsului, identificator unic pentru fiecare produs.
    - *Description*: Descrierea produsului (de exemplu, denumirea acestuia).
    - *Quantity*: Cantitatea produsului cumpărată de client.
    - *InvoiceDate*: Data emiterii facturii.
    - *UnitPrice*: Prețul unitar al produsului.
    - *CustomerID*: ID-ul clientului care a realizat achiziția.
    - *Country:* Țara clientului.
  - legături între structurile din cadrul aceleași surse de date.
    - Fiecare factură (identificată prin InvoiceNo) este asociată cu unul sau mai multe produse, iar produsele sunt identificate prin StockCode și Description.
    - Clienții sunt identificați prin CustomerID, iar fiecare client poate avea mai multe tranzacții în baza de date.

Implementare sursa de date externa

Fișierul **data.csv** l-am folosit și pentru creare unui graf cu noduri și relații între ele în **Neo4j.**



● **DS_3: properties.json**

Fișierul **properties.json** conține informații despre proprietăți imobiliare, extrase dintr-un fișier JSON. Aceste date sunt folosite pentru a analiza diferite proprietăți imobiliare (preț, număr de camere, dimensiune, etc.), pentru a înțelege piața imobiliară și comportamentele de achiziție. Datele includ detalii despre diverse tipuri de proprietăți, cum ar fi case și apartamente, disponibile în mai multe locații.

- Tip sursă de date
  - Tip model de date: document.
  - Tip format de acces: JSON.
- Descriere structuri de date
  - câmpuri/coloane/atribute etc.;

    price: Prețul proprietății (ex. 105000)

    bedrooms: Numărul de camere (ex. 3)

    bathrooms: Numărul de băi (ex. 2)

lot acres: Suprafața terenului (ex. 0.12)

address: Adresa completă a proprietății, incluzând:

- street: Strada (ex. "1962661.0")
- city: Orașul (ex. "Adjuntas")
- state: Statul (ex. "Puerto Rico")
- zip: Codul poștal (ex. "00601")

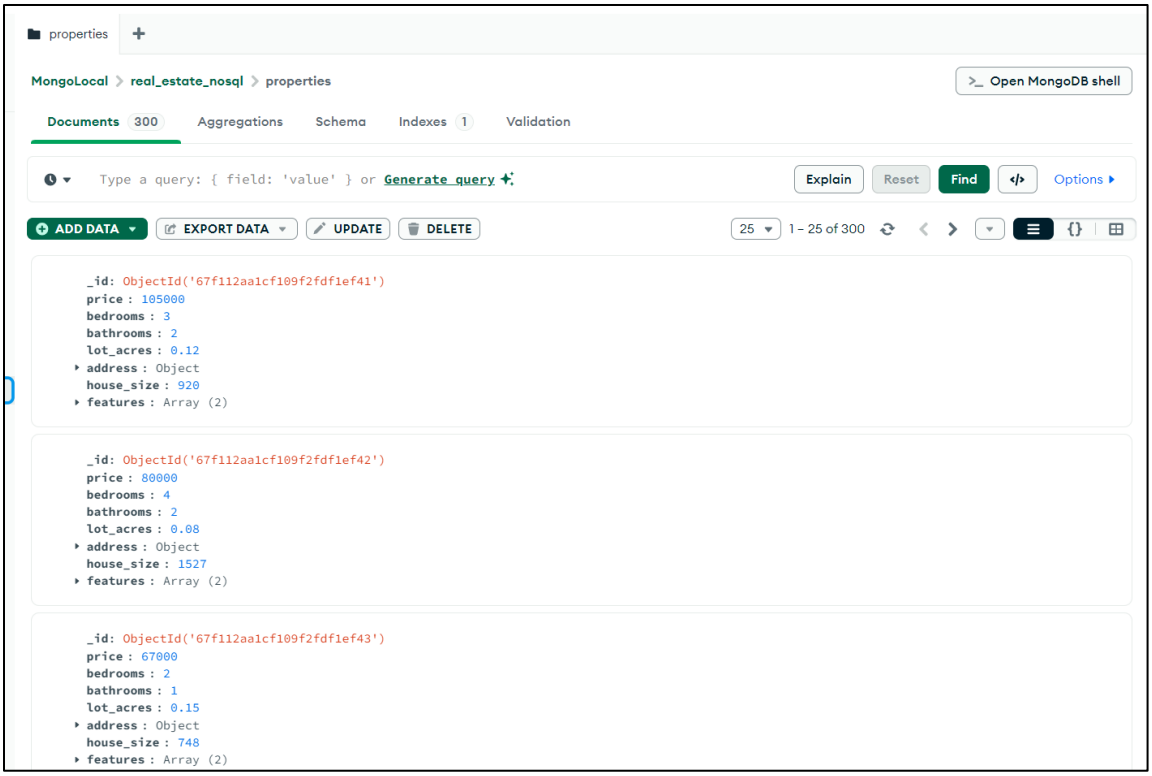
house_size: Suprafața casei în metri pătrați (ex. 920.0)

features: O listă cu caracteristici ale proprietății, cum ar fi (ex. garage, balcony)

- legături între structurile din cadrul aceleași surse de date.
  - Proprietăți - Locații: Fiecare proprietate poate fi legată de o locație printr-o relație LOCATED_IN.
  - Proprietăți - Tipuri de Proprietăți: Fiecare proprietate poate fi legată de un tip specific de proprietate (Casă, Apartament, etc.) printr-o relație IS_TYPE_OF.
  - Proprietăți - Tranzacții: Tranzacțiile de vânzare pot fi legate de proprietăți prin relația HAS_TRANSACTION.

○ Implementare sursa de date externa

Fișierul **properties.json** a fost importat într-o colecție MongoDB utilizând MongoDB Compass. Am creat o baza de date **real_estate_nosql** în care am creat o colecție **properties**. Această colecție stochează documentele ca obiecte JSON.



- **DS_4: customers.csv**

Fișierul **customers.csv** conține informații despre clienții care fac cereri pentru proprietăți iar fiecare rând reprezintă un client.

- Tip sursă de date
  - Tip model de date: relațional.
  - Tip format de access: CSV.
- Descriere structuri de date
  - câmpuri/coloane/atribute etc.;
    - id: Identificator unic al clientului (ex. 1, 2, 3, etc.).
    - customer_code: Codul unic al clientului (ex. C17850).
    - full_name: Numele complet al clientului (ex. "Client 17850").
    - email: Adresa de email a clientului (ex. "client17850@client.com").
    - phone: Numărul de telefon al clientului (ex. "+4070017850").
    - registration_date: Data la care clientul s-a înregistrat (ex. "2022-04-14").

- country: Țara clientului (ex. "United Kingdom").
- city: Orașul clientului (ex. "London").

- legături între structurile din cadrul aceleași surse de date.
  - id este legat de request_id din fișierul property_requests.csv, indicând că un client poate face mai multe cereri pentru proprietăți.
  - Fiecare client poate fi asociat unui property_id din fișierul properties.csv în cazul în care a achiziționat o proprietate sau a făcut o cerere pentru o proprietate.

○ Implementare sursa de date externa

■ Script_Comenzi_DDL_DML (pentru format SQL/NoSQL)

Acest script SQL creează tabela **customers** pentru a stoca informațiile despre clienți într-o bază de date relațională. Câmpurile sunt corespunzătoare celor din fișierul CSV (customer_code, full_name, email, etc.), iar fiecare client va avea un id unic, care va fi folosit pentru a realiza legături cu alte tabele.

```
7
8  -- CREARE TABELE
9  -- Tabela customers
10 CREATE TABLE customers (
11     id SERIAL PRIMARY KEY,
12     customer_code VARCHAR(10),
13     full_name VARCHAR(100),
14     email VARCHAR(100),
15     phone VARCHAR(20),
16     registration_date DATE,
17     country VARCHAR(50),
18     city VARCHAR(50)
19 );
```

DS_5: property_requests_pgadmin.csv

Fișierul **property_requests_pgadmin.csv** conține informații despre cererile făcute de clienți pentru diverse proprietăți iar fiecare rând reprezintă o cerere.

- Tip sursă de date
  - Tip model de date: relațional.

- Tip format de access: CSV.

- Descriere structuri de date
  - câmpuri/coloane/atribute etc.;
    - request_id: Identificatorul unic pentru cerere.
    - customer_id: Identificatorul clientului care face cererea.
    - request_date: Data la care cererea a fost făcută.
    - property_type: Tipul de proprietate solicitat (ex: Casă, Apartament, etc.).
    - max_budget: Bugetul maxim pentru achiziția proprietății.
    - min_surface: Suprafața minimă solicitată pentru proprietate.
    - preferred_city: Orașul preferat pentru proprietatea solicitată.
    - request_status: Statusul cererii (ex: "new", "matched", "closed").
  - legături între structurile din cadrul aceleași surse de date.
    - request_id este legat de customer_id, iar fiecare cerere este asociată unui client specific din fișierul customers.csv.
    - Cererile sunt asociate cu properties.csv pe baza tipului de proprietate dorit.

○ Implementare sursa de date externa

■ Script_Comenzi_DDL_DML (pentru format SQL/NoSQL)

Acest script SQL creează tabela property_requests, care va stoca cererile clienților pentru diverse tipuri de proprietăți. Câmpul customer_id face legătura între cererea de proprietate și clientul care a făcut cererea, prin referința la tabela customers.

```sql
-- Tabela property_requests
CREATE TABLE property_requests (
    request_id SERIAL PRIMARY KEY,
    customer_id INTEGER REFERENCES customers(id),
    request_date DATE,
    property_type VARCHAR(50),
    max_budget NUMERIC(12,2),
    min_surface INTEGER,
    preferred_city VARCHAR(50),
    request_status VARCHAR(20)
);
```

DS_6: request_feedback.csv

Fișierul **request_feedback.csv** conține feedback-ul dat de clienți pentru cereri de proprietăți. Fiecare rând reprezintă un feedback asociat unui request.

- Tip sursă de date
  - Tip model de date: relațional.
  - Tip format de access: CSV.
- Descriere structuri de date
  - câmpuri/coloane/atribute etc.;
    - id: Identificatorul unic pentru feedback.
    - request_id: Identificatorul cererii pentru care s-a dat feedback.
    - feedback_date: Data în care a fost dat feedback-ul.
    - rating: Scorul acordat cererii (ex: 3, 5, etc.).
    - comments: Comentarii adăugate de client pentru feedback-ul respectiv.
  - legături între structurile din cadrul aceleași surse de date.
    legătură indirectă între feedback și cererea originală (request) la care se face referire prin request_id. Feedback-ul pentru fiecare cerere este asociat unui request

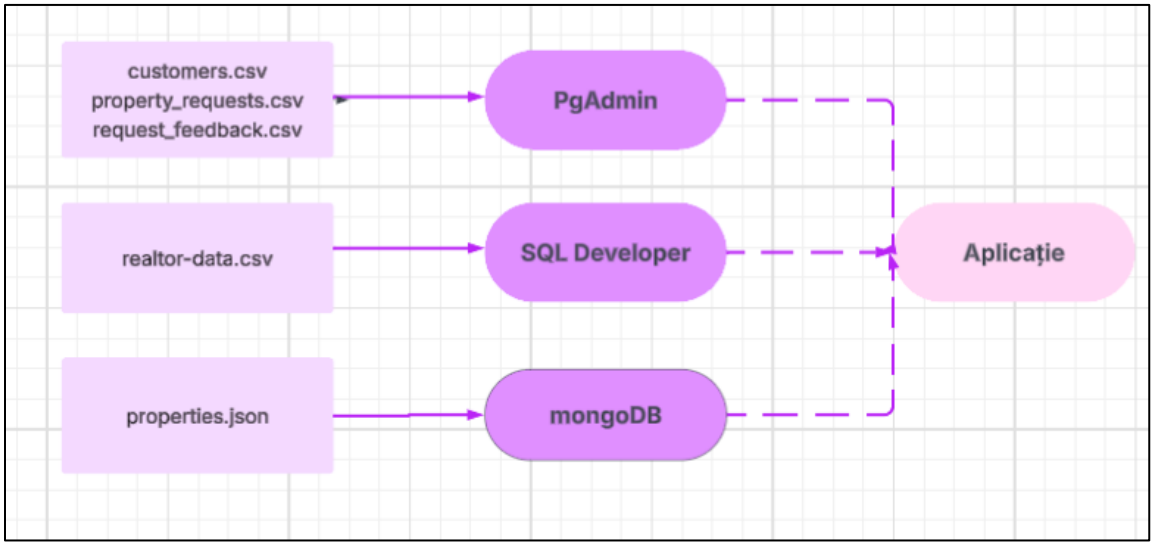
○ Implementare sursa de date externa

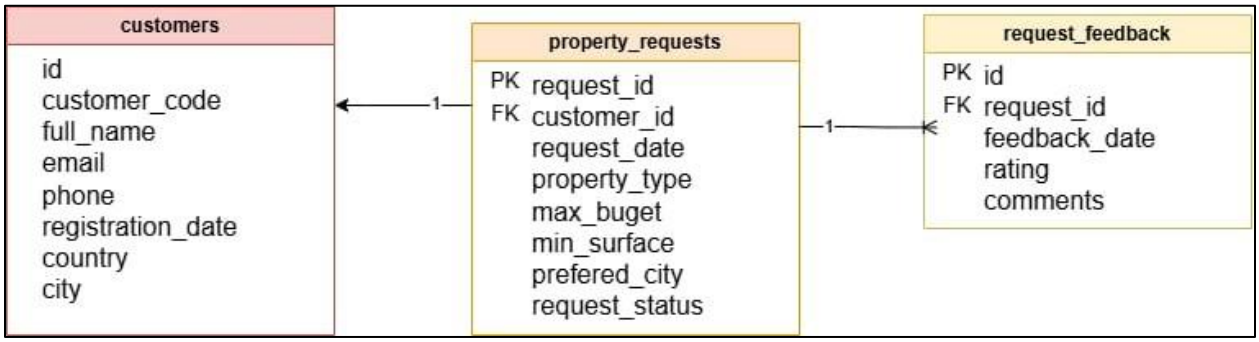
■ Script_Comenzi_DDL_DML (pentru format SQL/NoSQL)

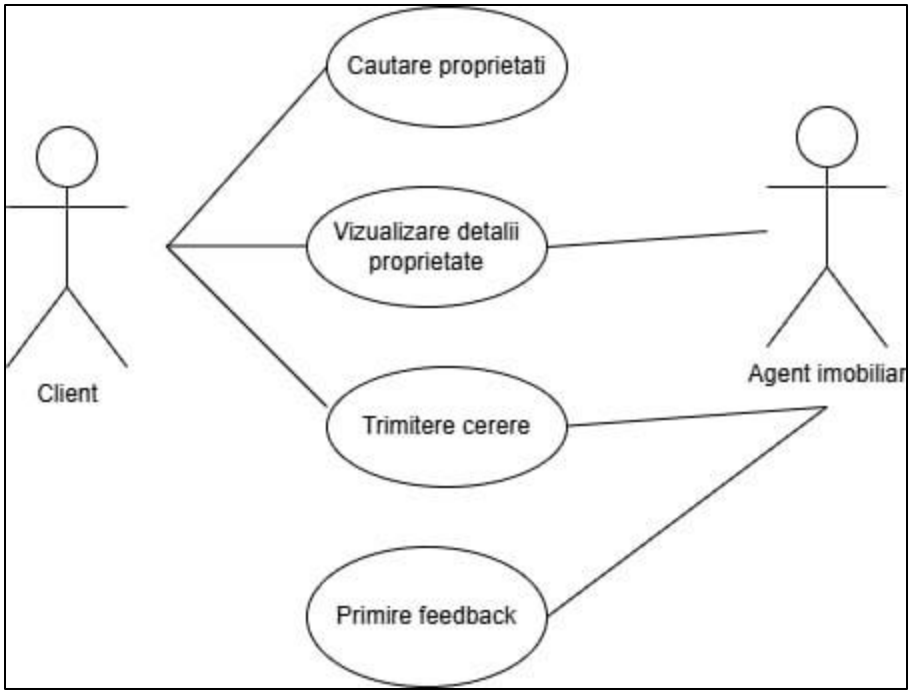
Acest script SQL creează tabela **request_feedback**, care va stoca feedback-ul clienților pentru fiecare cerere de proprietate. Câmpul request_id face legătura între feedback și cererea respectivă, iar câmpul rating permite evaluarea feedback-ului pe o scală de la 1 la 5.

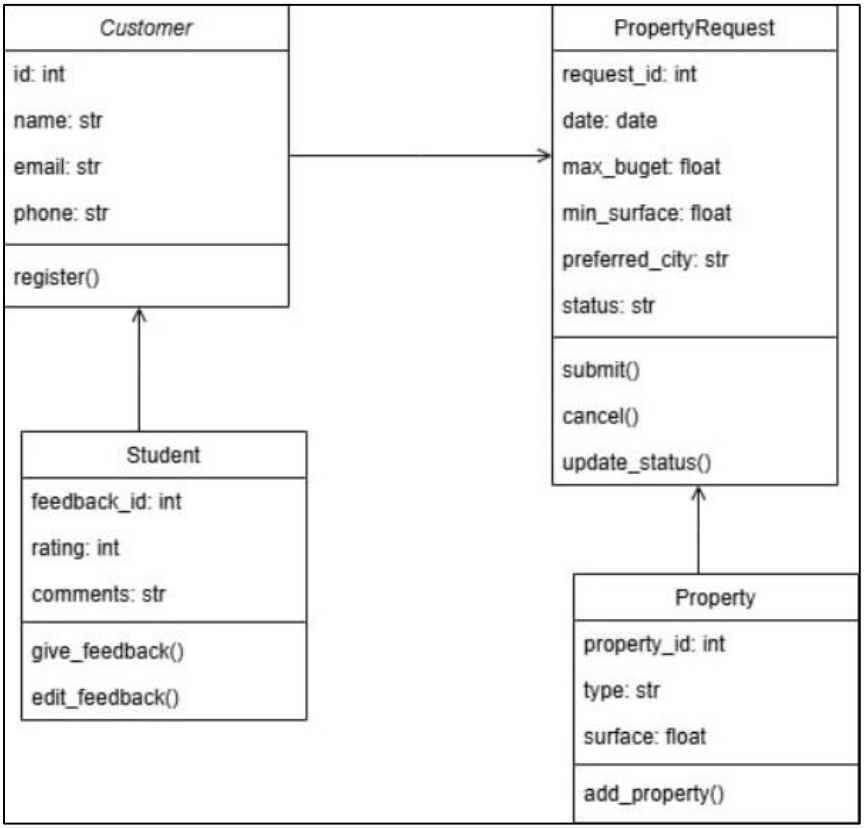
```
32
33  -- Tabela request_feedback
34  CREATE TABLE request_feedback (
35      id SERIAL PRIMARY KEY,
36      request_id INTEGER REFERENCES property_requests(request_id),
37      feedback_date DATE,
38      rating INTEGER CHECK (rating BETWEEN 1 AND 5),
39      comments TEXT
40  );
```

Diagrame






Cautare proprietati
Vizualizare detalii proprietate
Trimitere cerere
Primire feedback
Client
Agent imobiliar


Customer
id: int
name: str
email: str
phone: str
register()
PropertyRequest
request_id: int
date: date
max_buget: float
min_surface: float
preferred_city: str
status: str
submit()
cancel()
update_status()
Student
feedback_id: int
rating: int
comments: str
give_feedback()
edit_feedback()
Property
property_id: int
type: str
surface: float
add_property()

TEMA: L2. Arhitectura sistemului federativ de integrare [FDB Oracle]

IMPLEMENTARE ACCESS surse de date externe

DS_1: customers.csv

- Format/Tip de access: CSV
- Implementare acces la sursa de date externă:
  - Mecanism de acces: Am importat fișierul **customers.csv** într-o tabelă locală în pgAdmin utilizând interfața grafică pentru a crea tabela și pentru a importa fișierul CSV.
  - Definiție structură de access: Tabel local customers în baza de date SQL
- Implementare comenzi SQL:

```sql
-- CREARE TABELE
-- Tabela customers
CREATE TABLE customers (
    id SERIAL PRIMARY KEY,
    customer_code VARCHAR(10),
    full_name VARCHAR(100),
    email VARCHAR(100),
    phone VARCHAR(20),
    registration_date DATE,
    country VARCHAR(50),
    city VARCHAR(50)
);
```

DS_2: property_requests_pgadmin.csv

- Format/Tip de access: CSV
- Implementare acces la sursa de date externă:
  - Mecanism de acces: Am importat fișierul **property_requests_pgadmin.csv** într-o tabelă locală în pgAdmin utilizând interfața grafică pentru a crea tabela și pentru a importa fișierul CSV.
  - Definiție structură de access: Tabel local **property_requests** în baza de date SQL
- Implementare comenzi SQL:

```sql
-- Tabela property_requests
CREATE TABLE property_requests (
    request_id SERIAL PRIMARY KEY,
    customer_id INTEGER REFERENCES customers(id),
    request_date DATE,
    property_type VARCHAR(50),
    max_budget NUMERIC(12,2),
    min_surface INTEGER,
    preferred_city VARCHAR(50),
    request_status VARCHAR(20)
);
```

DS_3: request_feedback.csv

- Format/Tip de access: CSV
- Implementare acces la sursa de date externă:
  - Mecanism de acces: Am importat fișierul **request_feedback.csv** într-o tabelă locală în pgAdmin utilizând interfața grafică pentru a crea tabela și pentru a importa fișierul CSV.
  - Definiție structură de access: Tabel local **request_feedback** în baza de date SQL
- Implementare comenzi SQL:

```sql
-- Tabela request_feedback
CREATE TABLE request_feedback (
    id SERIAL PRIMARY KEY,
    request_id INTEGER REFERENCES property_requests(request_id),
    feedback_date DATE,
    rating INTEGER CHECK (rating BETWEEN 1 AND 5),
    comments TEXT
);
```

DS_4: properties.json

- Format/Tip de access: JSON
- Implementare acces la sursa de date externă:
  - Mecanism de acces: MongoDB Compass pentru importarea fișierului JSON într-o colecție MongoDB.
  - Definiție structură de access: Colectie locală în MongoDB numită **properties.**

DS_5: realtor-data.csv

- Format/Tip de access: CSV
- Implementare acces la sursa de date externă:
  - Mecanism de acces: Am importat manual date din fișierul **realtor-data.csv** în Oracle SQL Developer
  - Definiție structură de access: Tabele create în baza de date SQL
- Implementare comenzi SQL:

```
-- 1. Orașe
CREATE TABLE cities (
  id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  city_name VARCHAR2(100) NOT NULL,
  state VARCHAR2(50) NOT NULL,
  zip_code VARCHAR2(10),
  UNIQUE (city_name, state)
);

-- 2. Proprietari
CREATE TABLE owners (
  id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  full_name VARCHAR2(100) NOT NULL,
  email VARCHAR2(100),
  phone VARCHAR2(20),
  created_at DATE DEFAULT SYSDATE
);
```

DS_6: data.csv

- Format/Tip de access: CSV
- Implementare acces la sursa de date externă:
  - Mecanism de acces: Am importat manual date din fișierul **data.csv** în Neo4j pentru a crea noduri și a face relații între ele.
  - Definiție structură de access: Graf cu noduri.

TEMA: L3. Analytical Integration Model: Implementare OLAP views

1. Nivel CONSOLIDARE date

View_Consolidare_1: Consolidare Clienți și Cereri

Surse de date integrate: customers, property_requests

Definiție: View pentru joncționarea tabelor customers și property_requests pentru a obține informații despre cererile clienților, inclusiv tipul proprietății, data cererii și statusul acesteia.

```sql
-- View_Consolidare_1: Consolidare Clienți și Cereri
CREATE VIEW view_consolidare_customers_requests AS
SELECT
    c.full_name,
    c.email,
    c.city,
    r.property_type,
    r.request_date,
    r.request_status
FROM
    customers c
JOIN
    property_requests r ON c.id = r.customer_id;
```

Ca rezultat avem:

Data Output | Messages | Notifications

	full_name character varying (100)	email character varying (100)	city character varying (50)	property_type character varying (50)	request_date date	request_status character varying (20)
1	Client 17511	17511@client.com	United Kingdom	Studio	2025-03-01	Pending
2	Client 17924	17924@client.com	United Kingdom	Studio	2025-02-04	Pending
3	Client 15525	15525@client.com	United Kingdom	Apartment	2025-02-06	Open
4	Client 14237	14237@client.com	United Kingdom	Studio	2025-03-19	Closed
5	Client 17802	17802@client.com	United Kingdom	House	2025-03-08	Open
6	Client 16781	16781@client.com	United Kingdom	House	2025-02-02	Closed
7	Client 16725	16725@client.com	United Kingdom	House	2025-01-14	Closed
8	Client 14813	14813@client.com	United Kingdom	Apartment	2025-02-27	Pending
9	Client 15332	15332@client.com	Lithuania	Studio	2025-01-24	Pending
10	Client 15716	15716@client.com	United Kingdom	Apartment	2025-02-19	Open
11	Client 16638	16638@client.com	United Kingdom	Studio	2025-03-16	Open
12	Client 15660	15660@client.com	United Kingdom	House	2025-01-27	Closed
13	Client 12683	12683@client.com	France	Apartment	2025-01-03	Pending

View_Consolidare_2: Consolidare Cereri și Feedback

Surse de date integrate: property_requests, request_feedback

Definiție: Consolidare între tabelul property_requests și request_feedback pentru a adăuga feedback-ul la cererile de proprietăți.

```
-- View_Consolidare_2: Consolidare Cereri și Feedback
CREATE VIEW view_consolidare_requests_feedback AS
SELECT
    r.request_id,
    r.property_type,
    r.request_date,
    f.rating,
    f.comments
FROM
    property_requests r
JOIN
    request_feedback f ON r.request_id = f.request_id;
```

Ca rezultat avem:

Data Output | Messages | Notifications

	request_id integer	property_type character varying (50)	request_date date	rating integer	comments text
1	84	Studio	2025-01-02	1	Great deal
2	54	Studio	2025-01-10	1	Not satisfied
3	71	Studio	2025-02-20	3	Good experience
4	46	Studio	2025-03-06	5	Not satisfied
5	45	Apartment	2025-01-03	5	Good experience
6	40	Studio	2025-01-18	3	Great deal
7	23	Apartment	2025-02-28	4	Good experience
8	81	House	2025-03-22	3	Average
9	11	Apartment	2025-01-03	3	Good experience
10	1	Apartment	2025-02-08	1	Great deal

2. Schema analitică ROLAP (Tabele de fapte)

Tabela_de_fapte_1: Cereri și Feedback

Surse de date integrate: property_requests, request_feedback

Definiție: View pentru agregarea cererilor de proprietăți și feedback-ului asociat, calculând numărul de feedback-uri și media rating-ului.

```sql
-- Tabela_de_fapte_1: Cereri și Feedback
CREATE VIEW view_fact_requests_feedback AS
SELECT
    r.request_id,
    r.property_type,
    r.request_date,
    COUNT(f.id) AS feedback_count,
    ROUND(AVG(f.rating), 2) AS avg_rating
FROM
    property_requests r
LEFT JOIN
    request_feedback f ON r.request_id = f.request_id
GROUP BY
    r.request_id, r.property_type, r.request_date;
```

Ca rezultat avem:

Data Output | Messages | Notifications

	request_id integer	property_type character varying (50)	request_date date	feedback_count bigint	avg_rating numeric
1	55	Studio	2025-01-06	1	5.00
2	27	Studio	2025-01-02	1	1.00
3	23	Apartment	2025-02-28	1	4.00
4	56	Studio	2025-01-14	1	3.00
5	58	House	2025-01-11	0	[null]
6	91	House	2025-03-17	1	4.00
7	8	Apartment	2025-02-06	1	4.00
8	87	Apartment	2025-02-12	0	[null]
9	74	House	2025-02-17	1	1.00
10	29	Studio	2025-02-17	1	4.00
11	54	Studio	2025-01-10	1	1.00
12	71	Studio	2025-02-20	1	3.00
13	68	House	2025-02-21	1	4.00

Tabela_de_fapte_2: Buget și Cereri

Surse de date integrate: property_requests

Definiție: View pentru agregarea cererilor de proprietăți pe oraș și calcularea bugetului mediu solicitat.

```
-- Tabela_de_fapte_2: Buget și Cereri
CREATE VIEW view_fact_budget_requests AS
SELECT
    r.request_id,
    r.property_type,
    r.preferred_city,
    COUNT(r.request_id) AS total_requests,
    ROUND(AVG(r.max_budget), 2) AS avg_budget
FROM
    property_requests r
GROUP BY
    r.request_id, r.property_type, r.preferred_city;
```

Ca rezultat avem:

Data Output | Messages | Notifications

	request_id integer	property_type character varying (50)	preferred_city character varying (50)	total_requests bigint	avg_budget numeric
1	55	Studio	Greece	1	62918.00
2	27	Studio	RSA	1	99857.00
3	23	Apartment	Lebanon	1	123156.00
4	56	Studio	United Kingdom	1	112988.00
5	91	House	Australia	1	67626.00
6	58	House	Netherlands	1	62212.00
7	8	Apartment	Finland	1	103620.00
8	87	Apartment	Lithuania	1	121119.00
9	74	House	Denmark	1	43730.00

3. Tabele dimensionale OLAP

Tabela_dimensionala_1: Orașe

Surse de date integrate: customers, property_requests

Definiție: View dimensional pentru orașe, cu agregări ale cererilor și bugetului mediu pe oraș.

```
CREATE VIEW view_dim_city AS
SELECT
    c.city,
    COUNT(r.request_id) AS total_requests,
    ROUND(AVG(r.max_budget), 2) AS avg_budget
FROM
    customers c
JOIN
    property_requests r ON c.id = r.customer_id
GROUP BY
    c.city;
```

Ca rezultat avem:

Data Output | Messages | Notifications

	city character varying (50)	total_requests bigint	avg_budget numeric
1	Norway	1	52023.00
2	United Kingdom	92	100436.90
3	Germany	1	77597.00
4	Canada	1	149713.00
5	Lithuania	1	108888.00
6	Channel Islands	1	109608.00
7	Belgium	1	128713.00
8	France	2	113046.00

Tabela_dimensionala_2: Tipuri de Proprietăți

Surse de date integrate: property_requests

Definiție: View dimensional pentru tipurile de proprietăți, cu agregări ale cererilor și bugetului mediu solicitat.

```
CREATE VIEW view_dim_property_type AS
SELECT
    r.property_type,
    COUNT(r.request_id) AS total_requests,
    ROUND(AVG(r.max_budget), 2) AS avg_budget
FROM
    property_requests r
GROUP BY
    r.property_type;
```

Ca rezultat avem:

Data Output | Messages | Notifications

	property_type character varying (50)	total_requests bigint	avg_budget numeric
1	House	31	100226.39
2	Apartment	26	102474.42
3	Studio	43	100499.44

4. Tabele/view-uri cu agregări analitice OLAP

View_Analitic_OLAP_1: Rating și Feedback pe Orașe

Surse de date integrate: customers, request_feedback

Definiție: Calcularea rating-ului mediu pe orașe, împreună cu numărul de feedback-uri primite pentru fiecare oraș.

```sql
CREATE VIEW view_analitic_olap_rating_per_property_type AS
SELECT
    r.property_type,
    COUNT(f.id) AS feedback_count,
    ROUND(AVG(f.rating), 2) AS avg_rating
FROM
    property_requests r
JOIN
    request_feedback f ON r.request_id = f.request_id
GROUP BY
    r.property_type;
```

Ca rezultat avem:

Data Output | Messages | Notifications

	property_type character varying (50)	feedback_count bigint	avg_rating numeric
1	House	21	2.76
2	Apartment	20	2.70
3	Studio	29	3.14

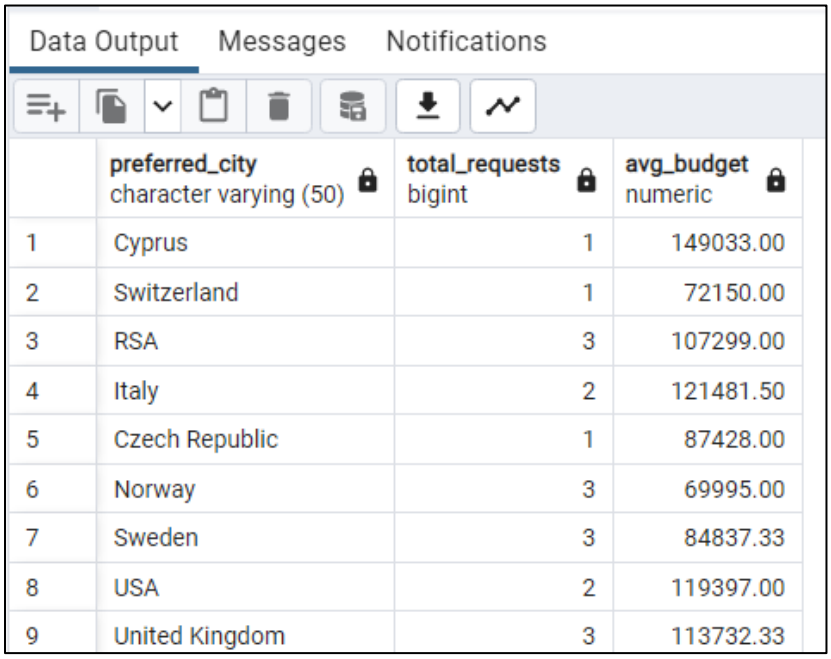
View_Analitic_OLAP_2: Buget și Cereri pe Orașe

Surse de date integrate: property_requests

Definiție: Calcularea bugetului mediu solicitat pentru fiecare oraș și numărul total de cereri pentru fiecare oraș.

```sql
-- View_Analitic_OLAP_1: Buget și Cereri pe Orașe
CREATE VIEW view_analitic_olap_budget_per_city AS
SELECT
    r.preferred_city,
    COUNT(r.request_id) AS total_requests,
    ROUND(AVG(r.max_budget), 2) AS avg_budget
FROM
    property_requests r
GROUP BY
    r.preferred_city;
```

Ca rezultat avem:



Data Output | Messages | Notifications

	preferred_city character varying (50)	total_requests bigint	avg_budget numeric
1	Cyprus	1	149033.00
2	Switzerland	1	72150.00
3	RSA	3	107299.00
4	Italy	2	121481.50
5	Czech Republic	1	87428.00
6	Norway	3	69995.00
7	Sweden	3	84837.33
8	USA	2	119397.00
9	United Kingdom	3	113732.33

View_Analitic_OLAP_3: Rating pe Tipuri de Proprietăți

Surse de date integrate: property_requests, request_feedback

Definiție: Analiza rating-ului mediu pe tipurile de proprietăți, împreună cu numărul de feedback-uri pentru fiecare tip de proprietate.

```
CREATE VIEW view_analitic_olap_rating_per_property_type AS
SELECT
    r.property_type,
    COUNT(f.id) AS feedback_count,
    ROUND(AVG(f.rating), 2) AS avg_rating
FROM
    property_requests r
JOIN
    request_feedback f ON r.request_id = f.request_id
GROUP BY
    r.property_type;
```

5. Funcții OLAP pentru CUBE și ROLLUP

CUBE: combinații posibile oraș + rating

Surse de date integrate: customers, request_feedback

Definiție: Utilizarea funcției CUBE pentru a analiza combinațiile posibile între oraș și rating.

```
CREATE VIEW view_cube_city_rating AS
SELECT
    c.city,
    f.rating,
    COUNT(*) AS total_feedback,
    ROUND(AVG(f.rating), 2) AS avg_rating
FROM
    customers c
JOIN
    property_requests r ON c.id = r.customer_id
JOIN
    request_feedback f ON r.request_id = f.request_id
GROUP BY
    CUBE(c.city, f.rating);
```

ROLLUP: totaluri cumulative pe oraș + tip proprietate

Surse de date integrate: property_requests

Definiție: Utilizarea funcției ROLLUP pentru a obține totaluri cumulative pe oraș și tip proprietate.

```
CREATE VIEW view_rollup_city_property_type AS
SELECT
    r.preferred_city,
    r.property_type,
    COUNT(*) AS total_requests,
    ROUND(AVG(r.max_budget), 2) AS avg_budget
FROM
    property_requests r
GROUP BY
    ROLLUP(r.preferred_city, r.property_type);
```

TEMA: P4. REST and Web Model

1. Resursa RESTful: Conectarea la baza de date cu PostgREST

Surse de date integrate:

- DS_x + DS_y + DS_z: **customers.csv**, **property_requests.csv**, **request_feedback.csv** (tabele din PostgreSQL)

Descriere:

- Am implementat un API RESTful utilizând **PostgREST**, care expune datele din baza de date PostgreSQL. Aceste date sunt accesibile prin endpoint-uri REST, iar răspunsul este furnizat în format JSON.

Implementare acces la sursa de date externă:

- Am configurat PostgREST pentru a expune tabelele din pgAdmin printr-un API REST. Astfel, am accesat sursele de date externe (CSV importate în PostgreSQL).

Pasul 1: Am importat fișierele CSV în PostgreSQL folosind pgAdmin.

Pasul 2: Am creat view-uri în PostgreSQL pentru a structura datele ce vor fi expuse prin API.
Pasul 3: Am configurat PostgREST pentru a accesa aceste view-uri și a le expune prin API.

2. Comenzi SQL utilizate pentru crearea view-urilor în PostgreSQL:

view pentru customers:

```sql
-- Tabela customers
CREATE TABLE customers (
    id SERIAL PRIMARY KEY,
    customer_code VARCHAR(10),
    full_name VARCHAR(100),
    email VARCHAR(100),
    phone VARCHAR(20),
    registration_date DATE,
    country VARCHAR(50),
    city VARCHAR(50)
);
```

property_requests:

```sql
-- Tabela property_requests
CREATE TABLE property_requests (
    request_id SERIAL PRIMARY KEY,
    customer_id INTEGER REFERENCES customers(id),
    request_date DATE,
    property_type VARCHAR(50),
    max_budget NUMERIC(12,2),
    min_surface INTEGER,
    preferred_city VARCHAR(50),
    request_status VARCHAR(20)
);
```

request_feedback:

Aceste view-uri vor fi expuse ca endpoint-uri REST prin PostgREST.

3. Configurare PostgREST:

Fișierul de configurare pentru PostgREST: În acest fișier configurăm conexiunea la baza de date PostgreSQL și definim view-urile ca endpoint-uri REST.

```
db-uri = "postgres://postgres:Claudia*100@localhost:5433/real_estate_pg"

db-schema = "public"

db-anon-role = "anon"
```

Pasul 4: În fișierul de configurare, am definit URL-ul bazei de date, schema utilizată și rolul anonim pentru accesul public la date. Acest lucru permite expunerea datelor prin API REST.

4. Accesul la date prin API PostgREST:

După configurarea PostgREST, am creat un server care rulează pe portul 3000, expunând datele la următoarele endpoint-uri:

Pentru clienți (customers): URL-ul: **http://localhost:3000/customers**

Răspuns (JSON):


```
[
  {
    "id": 1,
    "customer_code": "C17850.0",
    "full_name": "Client 17850",
    "email": "17850@client.com",
    "phone": "4070017850",
    "registration_date": "2022-02-23",
    "country": "United Kingdom",
    "city": "United Kingdom"
  },
  {
    "id": 2,
    "customer_code": "C13047.0",
    "full_name": "Client 13047",
    "email": "13047@client.com",
    "phone": "4070013047",
    "registration_date": "2022-12-25",
    "country": "United Kingdom",
    "city": "United Kingdom"
  },
  {
```


Pentru cereri de proprietăți (property_requests): URL: http://localhost:3000/property_requests

Răspuns (JSON):


```
[
  {
    "request_id": 1,
    "customer_id": 3085,
    "request_date": "2025-02-08",
    "property_type": "Apartment",
    "max_budget": 140902,
    "min_surface": 43,
    "preferred_city": "Greece",
    "request_status": "Open"
  },
  {
    "request_id": 2,
    "customer_id": 1701,
    "request_date": "2025-03-29",
    "property_type": "Studio",
    "max_budget": 139102,
    "min_surface": 113,
    "preferred_city": "Saudi Arabia"
    "request_status": "Pending"
  },
  {
```

Pentru feedback (request_feedback): URL-ul: http://localhost:3000/request_feedback

Răspuns (JSON):


```
[
  {
    "id": 1,
    "request_id": 84,
    "feedback_date": "2025-01-12",
    "rating": 1,
    "comments": "Great deal"
  },
  {
    "id": 2,
    "request_id": 54,
    "feedback_date": "2025-01-22",
    "rating": 1,
    "comments": "Not satisfied"
  },
  {
    "id": 3,
    "request_id": 71,
    "feedback_date": "2025-02-28",
    "rating": 3,
    "comments": "Good experience"
```


View-uri

View : Integrare clienți +cereri+feedback

URL-ul: **http://localhost:3000/v_request_feedback_summary**

```sql
-- View 1: Integrare clienți + cereri + feedback
CREATE OR REPLACE VIEW v_request_feedback_summary AS
SELECT
  c.full_name,
  c.city AS customer_city,
  r.request_id,
  r.property_type,
  r.max_budget,
  r.request_date,
  f.feedback_date,
  f.rating,
  f.comments
FROM
  customers c
JOIN
  property_requests r ON c.id = r.customer_id
LEFT JOIN
  request_feedback f ON r.request_id = f.request_id;
```

Răspuns (JSON):


```
{
  "full_name": "Client 17511",
  "customer_city": "United Kingdom",
  "request_id": 50,
  "property_type": "Studio",
  "max_budget": 123934,
  "request_date": "2025-03-01",
  "feedback_date": "2025-03-17",
  "rating": 3,
  "comments": "Good experience"
},
{
  "full_name": "Client 17924",
  "customer_city": "United Kingdom",
  "request_id": 69,
  "property_type": "Studio",
  "max_budget": 132286,
  "request_date": "2025-02-04",
  "feedback_date": "2025-02-19",
  "rating": 5,
  "comments": "Good experience"
},
```


View : Statistici pe oraș

URL-ul: **http://localhost:3000/v_city_stats**

```sql
CREATE OR REPLACE VIEW v_city_stats AS
SELECT
  r.preferred_city,
  COUNT(*) AS total_requests,
  ROUND(AVG(r.max_budget), 2) AS avg_budget,
  COUNT(f.id) AS feedback_count,
  ROUND(AVG(f.rating), 2) AS avg_rating
FROM property_requests r
LEFT JOIN request_feedback f ON r.request_id = f.request_id
GROUP BY r.preferred_city;
```

Răspuns (JSON):


```
[
  {
    "preferred_city": "Cyprus",
    "total_requests": 1,
    "avg_budget": 149033,
    "feedback_count": 1,
    "avg_rating": 3
  },
  {
    "preferred_city": "Switzerland",
    "total_requests": 1,
    "avg_budget": 72150,
    "feedback_count": 1,
    "avg_rating": 3
  },
  {
    "preferred_city": "Italy",
    "total_requests": 2,
    "avg_budget": 121481.5,
    "feedback_count": 1,
    "avg_rating": 4
```

View : Clienți cu scor slab și comentarii

URL-ul: **http://localhost:3000/v_negative_feedback_clients**

```
-- View 3: Clienți cu scor slab și comentarii
CREATE OR REPLACE VIEW v_negative_feedback_clients AS
SELECT c.full_name, c.email, f.rating, f.comments
FROM customers c
JOIN property_requests r ON c.id = r.customer_id
JOIN request_feedback f ON r.request_id = f.request_id
WHERE f.rating <= 2;
```

Răspuns (JSON):


```
[
  {
    "full_name": "Client 15839",
    "email": "15839@client.com",
    "rating": 1,
    "comments": "Great deal"
  },
  {
    "full_name": "Client 14237",
    "email": "14237@client.com",
    "rating": 2,
    "comments": "Great deal"
  },
  {
    "full_name": "Client 14813",
    "email": "14813@client.com",
    "rating": 1,
    "comments": "Could be better"
  },
  {
    "full_name": "Client 16668",
    "email": "16668@client.com",
    "rating": 2,
    "comments": "Great deal"
  },
```


5. Testarea și accesul API:

Testarea API-ului:

Am folosit comanda curl pentru a testa fiecare endpoint:

curl http://localhost:3000/customers

curl http://localhost:3000/property_requests

curl http://localhost:3000/request_feedback

Partea APEX și Docker

Partea de implementare a interfeței web folosind APEX nu a fost realizată din cauza unor dificultăți tehnice legate de mediul de dezvoltare. În timpul procesului de configurare și rulare a aplicației APEX, aceasta s-a blocat din cauza unor limitări de memorie. De asemenea, am întâmpinat probleme legate de setările Docker, care au afectat performanța și au dus la blocarea aplicației.

Am configurat Docker pentru a crea un mediu izolat pentru baza de date și interfețele necesare, dar din cauza resurselor limitate ale sistemului, Docker a întâmpinat probleme în rularea aplicației APEX și nu am reușit să finalizez secțiunea respectivă.

Am reușit totuși să lucrez la toate celelalte părți ale proiectului, inclusiv importul și manipularea datelor în PostgreSQL și realizarea corectă a interogărilor SQL pentru analiza datelor. Din păcate, din cauza problemelor tehnice legate de memoria sistemului și configurația Docker, implementarea completă a aplicației APEX nu a putut fi finalizată.

MongoDB COMPASS - Agregări

Am folosit MongoDB pentru a stoca datele din fișierul **properties.json,** pe care l-am importat într-o colecție numită properties. În cadrul acestei implementări, am realizat și agregări pentru a extrage informații utile din documentele din colecția respectivă.

Detalii despre colecția properties Fișierul **properties.json** conține informații despre diverse proprietăți, inclusiv prețuri, tipuri de proprietăți, locații și alte caracteristici relevante.

Datele au fost structurate sub formă de documente JSON și au fost importate în MongoDB utilizând funcționalitatea de import a acestuia.

Agregări realizate

Am efectuat mai multe agregări pe colecția properties pentru a obține statistici și date analitice utile, cum ar fi prețurile medii ale proprietăților, numărul de proprietăți disponibile în diverse locații sau tipuri de proprietăți. Exemple de agregări realizate:

Agregare: Calculeaza prețul mediu al proprietăților pentru fiecare oraș

```
//Calculează prețul mediu al proprietăților pentru fiecare oraș
{
  _id: "$address.city",
  avg_price: {
    $avg: "$price"
  },
  total_bedrooms: {
    $sum: "$bedrooms"
  },
  max_house_size: {
    $max: "$house_size"
  }
}
```

Rezultat

Output after $group stage (Sample of 10 documents)

```
_id: "Morovis"
avg_price : 287500
total_bedrooms : 8
max_house_size : 2500
```

```
_id: "Corozal"
avg_price : 80000
total_bedrooms : 4
max_house_size : 1500
```

Agregare: Prețul mediu al proprietăților, grupate după numărul de dormitoare

```
//Prețul mediu al proprietăților, grupate după numărul de dormitoare (be
{
  _id: "$bedrooms",
  average_price: {
    $avg: "$price"
  },
  total_properties: {
    $sum: 1
  }
}
```

Stage 2 $project

```
{
  bedrooms: "$_id",
  average_price: 1,
  total_properties: 1,
  _id: 0
}
```

Rezultat

Agregare: Preț mediu pe metru pătrat grupat după oraș și anul construcției


{
_id: {
city: "$address.city",
year_built: "$year_built"
},
avg_price_per_sqm: {
$avg: "$price_per_sqm"
},
total_properties: {
$sum: 1
}
}
Stage 4 $project
{
city: "$_id.city",
year_built: "$_id.year_built",
avg_price_per_sqm: 1,
total_properties: 1,
_id: 0
}


Stage 5 $sort
{
city: 1,
year_built: 1
}

Rezultat


avg_price_per_sqm : 83.2603712878335
total_properties : 2
city : "Adjuntas"
avg_price_per_sqm : 237.19928977740753
total_properties : 22
city : "Aguada"