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: PostgreSQL (PGAdmin)

Tabele: customers, property_requests, request_feedback

- Tip sursă de date
  - Tip model de date: relațional.
  - Tip format de acces: SQL.
- Descriere structuri de date
  - câmpuri/coloane/atribute etc.;
    - *Tabel customers*:
      - id
      - customer_code
      - full_name
      - email
      - phone
      - registration_date
      - country
      - city
    - *Tabel property_requests:*
      - request_id
      - customer_id
      - request_date
      - property_type
      - max_budget
      - min_surface
      - preferred_city

- request_status

- *Tabel request_feedback:*

  - id, request_id
  - feedback_date
  - rating
  - comments

- Legături:
  - property_requests.customer_id → customers.id
  - request_feedback.request_id → property_requests.request_id

- Implementare:

- 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 );
```

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)
);
```

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.

```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_2: Oracle SQL Developer (Schema agent_imobiliar)

Tabele: cities, owners, property_types, real_estate

- Tip sursă de date
  - Tip model de date: relațional.
  - Tip format de acces: SQL.
- Descriere structuri de date
  - câmpuri/coloane/atribute etc.;
    - *Tabel cities:*
      - id,
      - city_name
      - state
      - zip_code

- *Tabel owners:*
    - *id,*
    - *full_name,*
    - *email,*
    - *phone,*
    - *created_at*
- *Tabel property_types:*
    - *id*
    - *type_name*
    - *description*
- *Tabel real_estate:*
    - id, price
    - beds
    - baths
    - house_size
    - listed_date
    - is_available
    - city_id
    - property_type_id
    - owner_id
- Legături:
    - real_estate.city_id → cities.id
    - real_estate.property_type_id → property_types.id
    - real_estate.owner_id → owners.id

- Implementare:
- Script_Comenzi_DDL_DML (pentru format SQL/NoSQL)

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

- Tabelele cities și owners

```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
);
```

- Tabelele property_types și real_estate

```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');
```

```
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);
```

- **DS_3: MongoDB (real_estate DB)**

  Colecții: market_insights, property_history, property_reviews, site_activity_logs

  - Tip sursă de date
    - Tip model de date: documente(MongoDB)
    - Tip format de acces: JSON (prin RESTHeart)
  - Descriere structuri de date
    - câmpuri/coloane/atribute etc.;
      - *market_insights:*
        - {_id, city, average_price, average_surface, timestamp, trend, oracle_id}
      - *property_history:*
        - {_id, property_id, previous_price, new_price, modification_date, reason, oracle_id}
      - *property_reviews:*
        - {_id, property_id, review_date, rating, reviewer, comment, oracle_id}
      - *site_activity_logs:*
        - {_id, user_id, action, timestamp, ip_address, oracle_id}

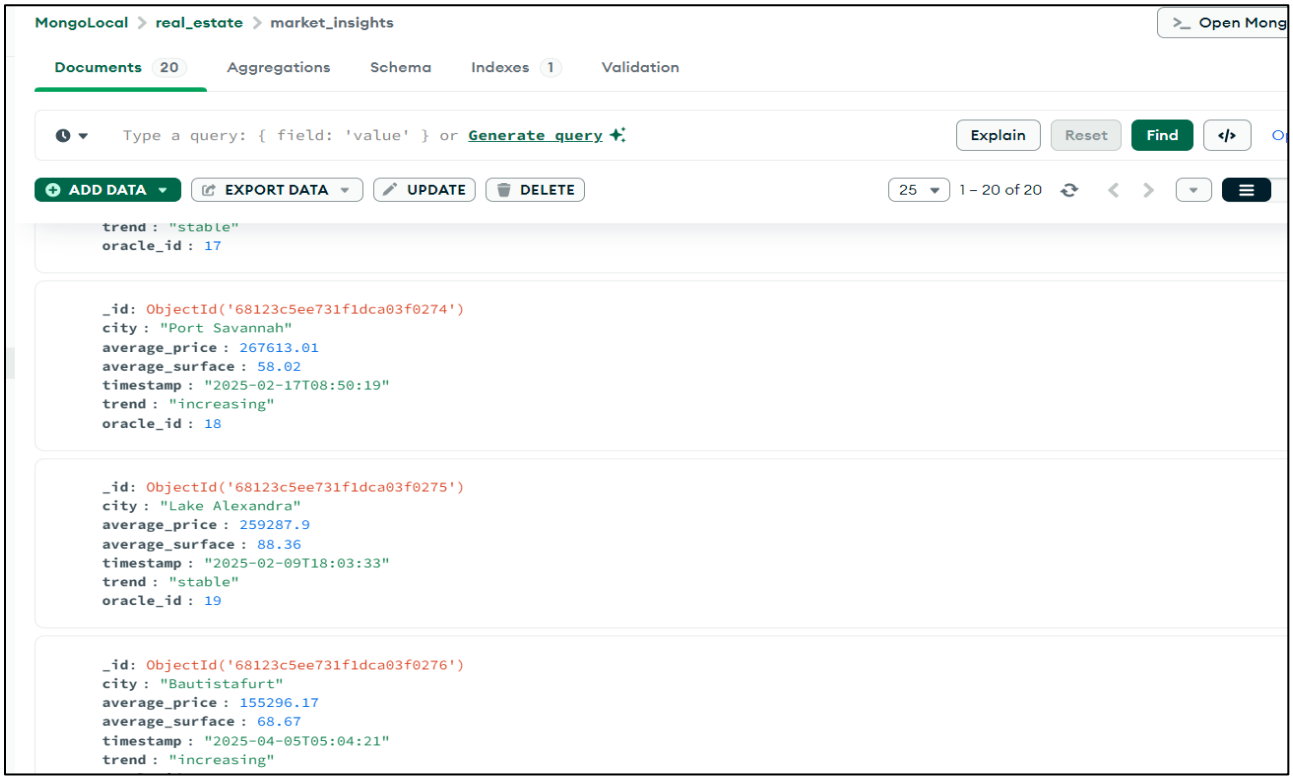
- Legături:
  - oracle_id din documente face legătura cu id din tabele Oracle (real_estate sau customers)

- Implementare:

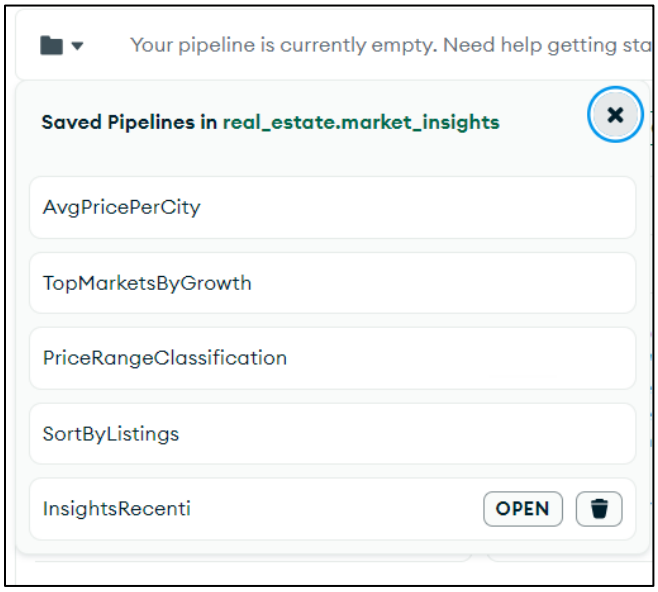
  - JSON export cu câteva documente / capturi MongoDB Compass

În MongoDB Compass am creat colecțiile:

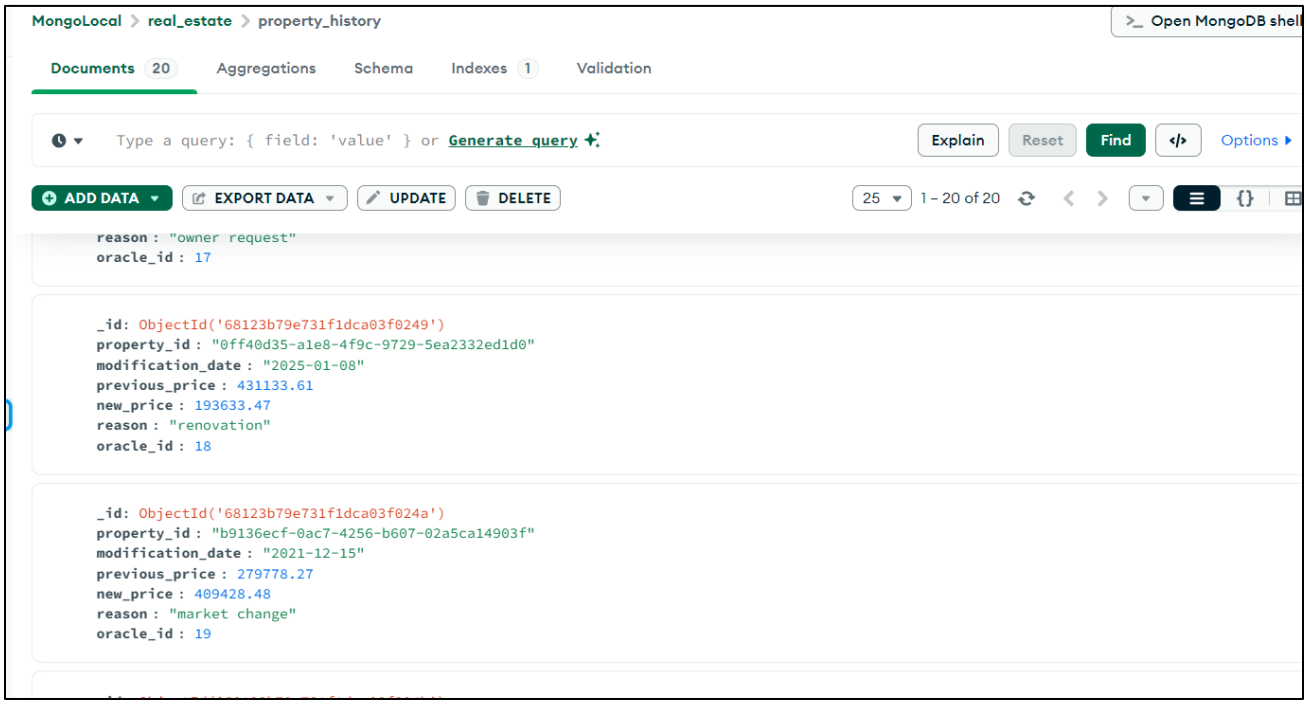
- Colecția **market_insights**



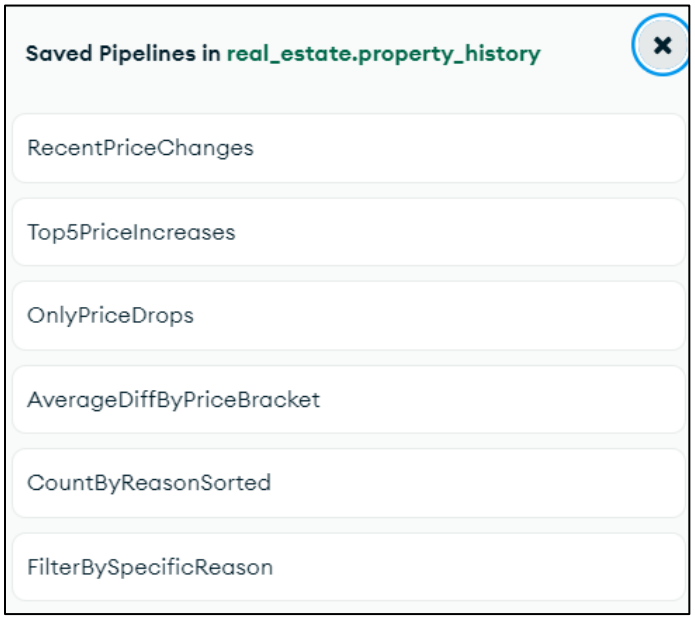
- Agregarile pentru colecția **market_insights**



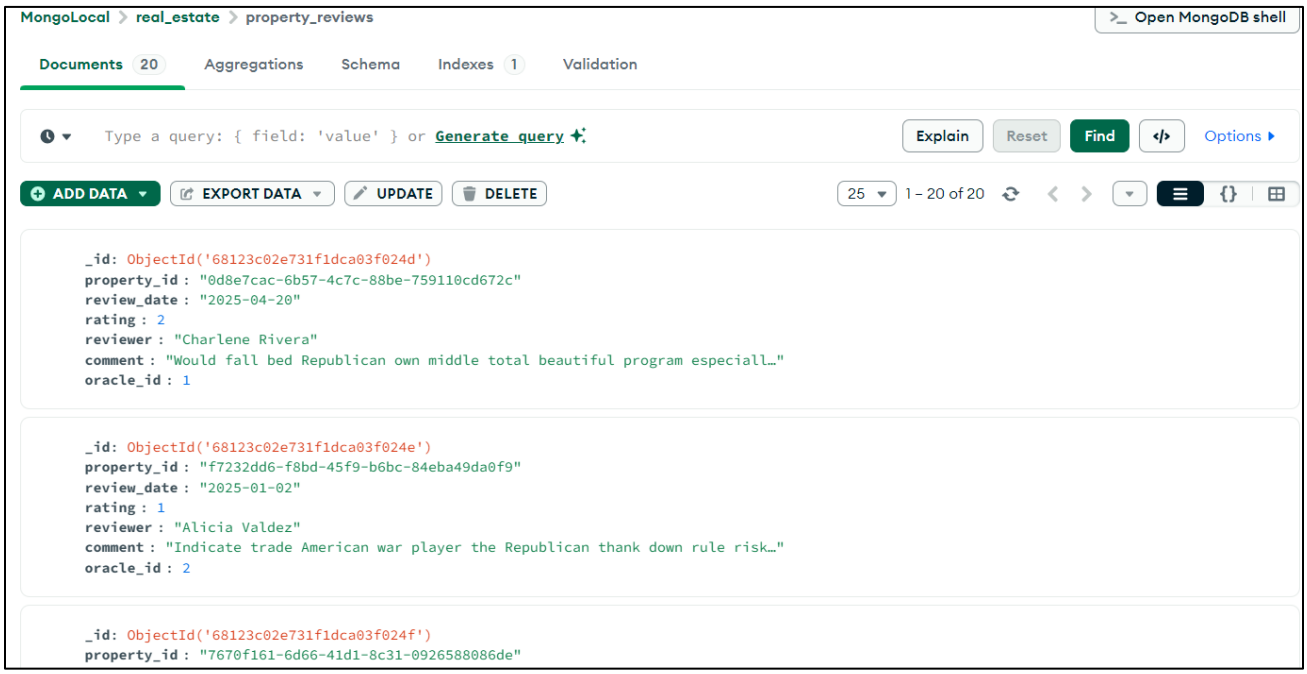
- Colecția **property_history**



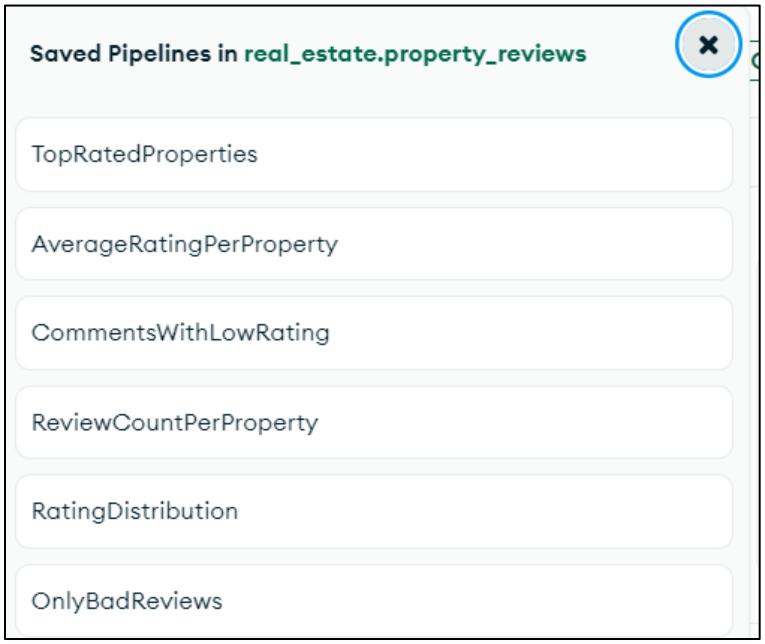
- Agregarile pentru colecția **property_history**



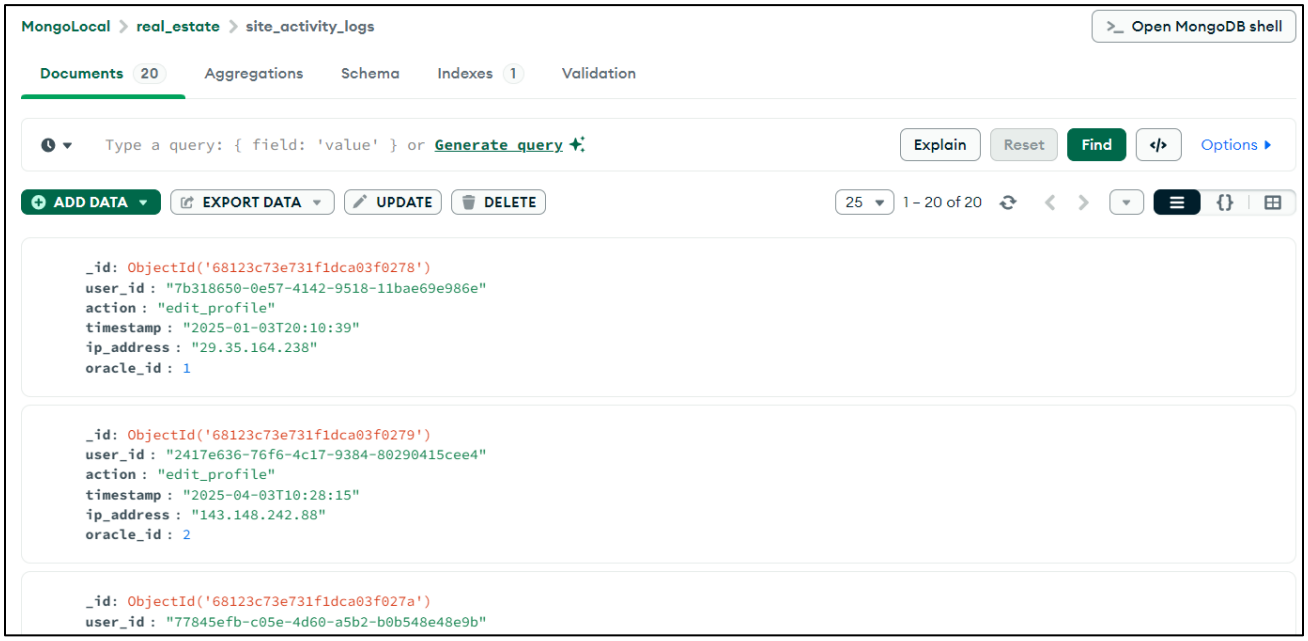
- Colecția **property_reviews**



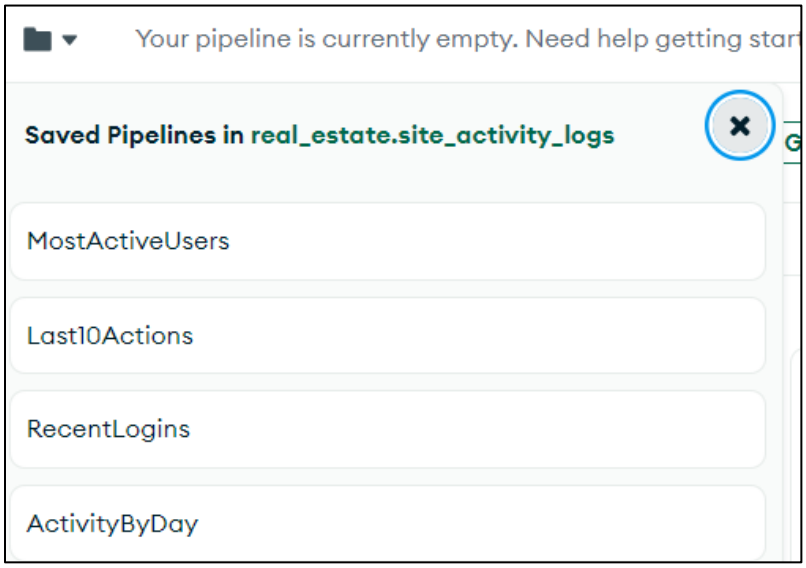
- Agregarile pentru colecția **property_reviews**



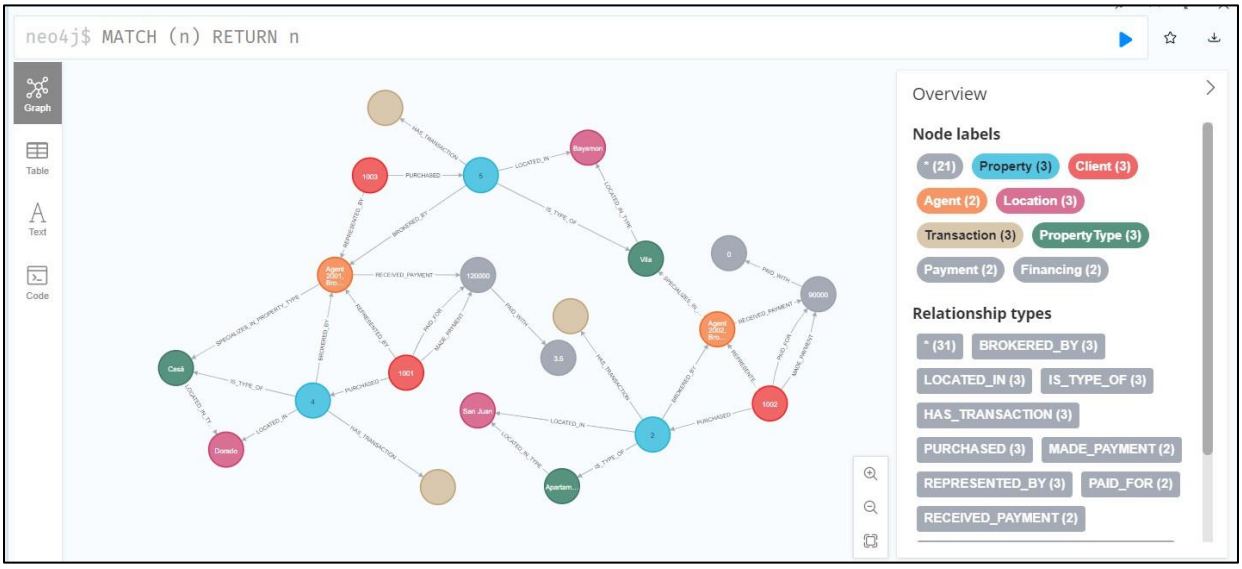
- Colecția **site_activity_logs**



- Agregarile pentru colecția **site_activity_logs**

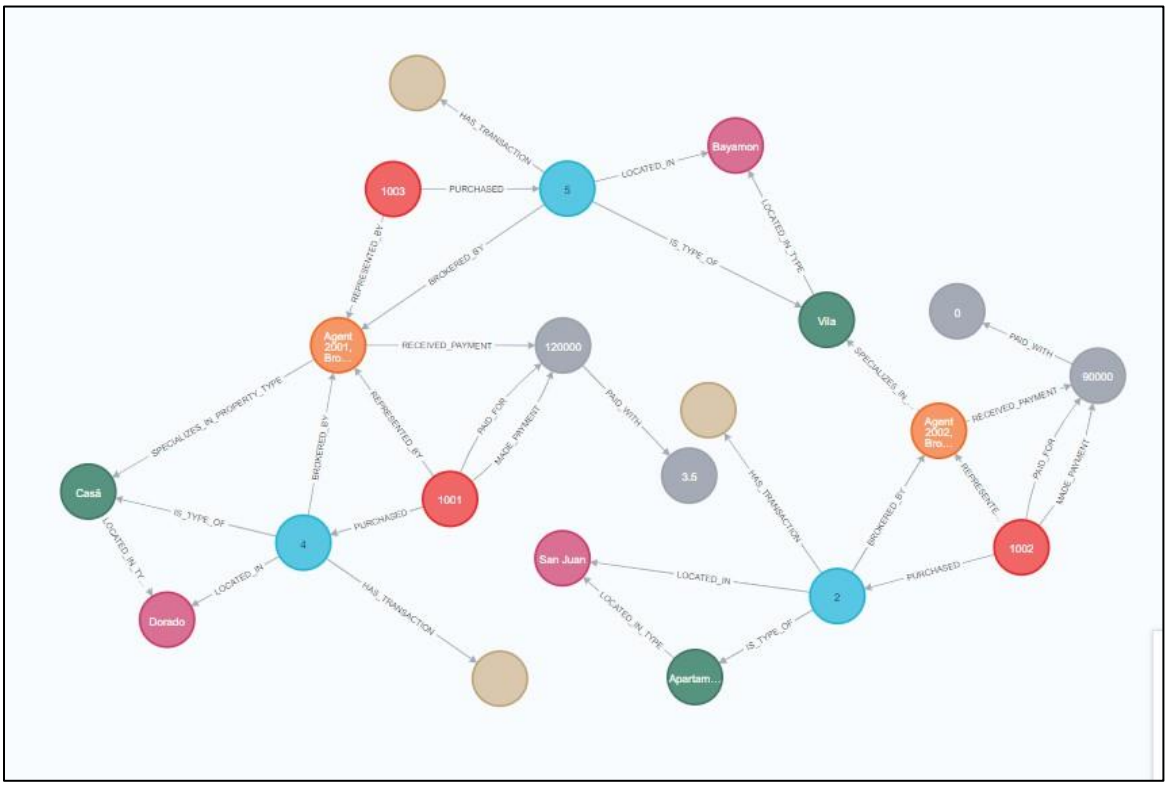


Fișierul **realtor-data.csv** l-am folosit și pentru crearea unui graf cu noduri și relații între ele în **Neo4j.**

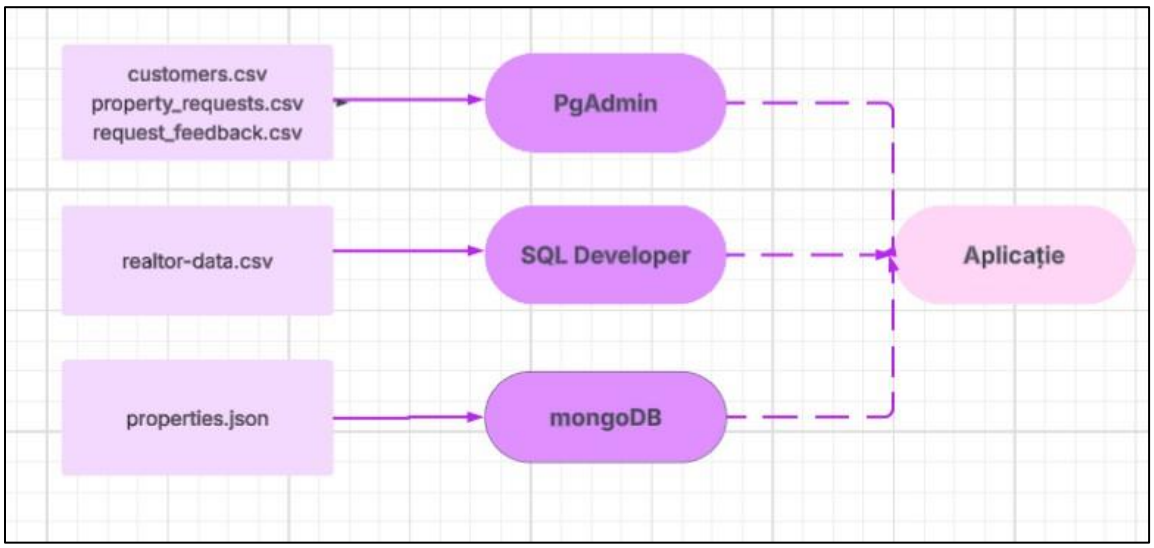


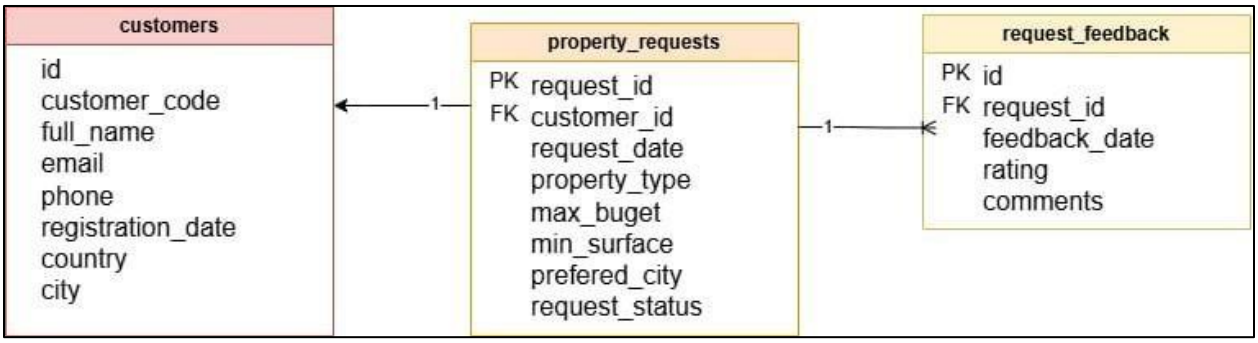
Fișierul **data.csv** conține informații despre facturile emise, detalii ale produselor vândute și clienților care au efectuat achiziții. Acesta poate fi utilizat pentru analiza vânzărilor, comportamentului clienților și performanței produselor.

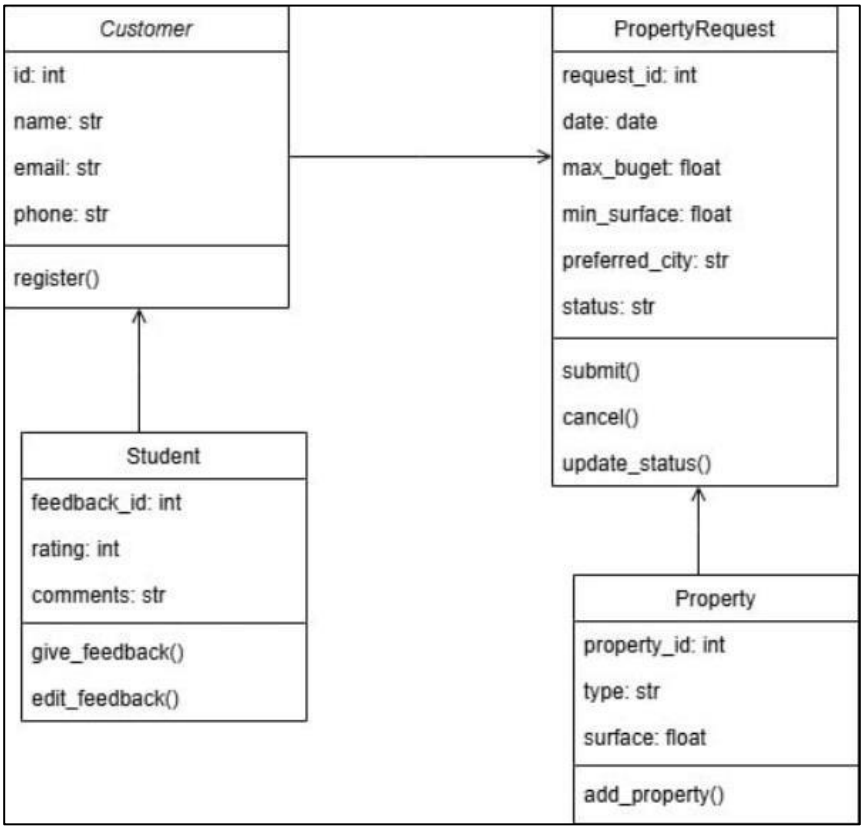
Fișierul **data.csv** l-am folosit și pentru creare unui graf cu noduri și relații între ele în **Neo4j.**

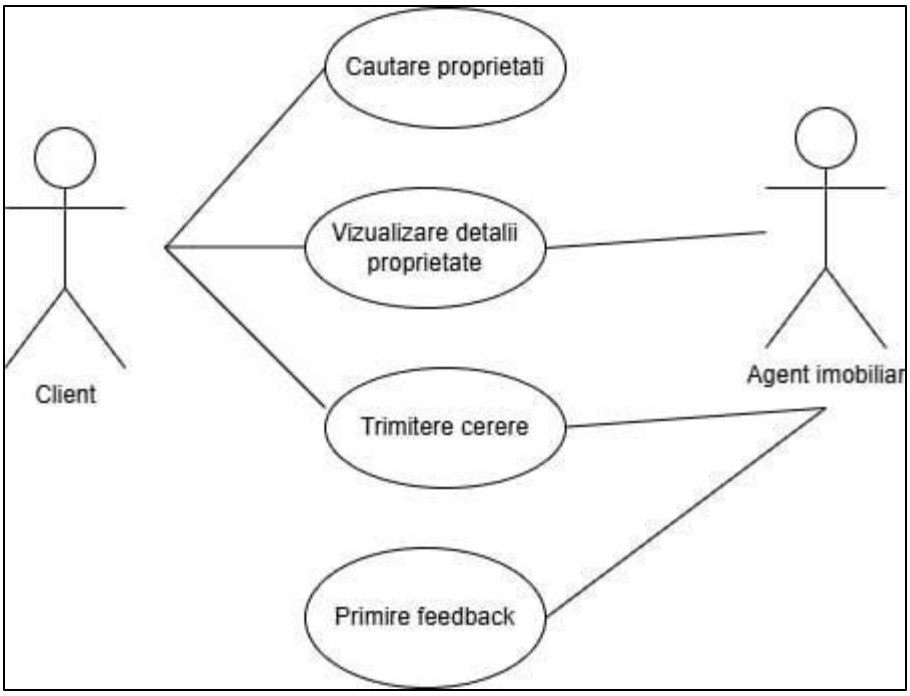


DIAGRAME




customers
id
customer_code
full_name
email
phone
registration_date
country
city
1
property_requests
PK request_id
FK customer_id
request_date
property_type
max_buget
min_surface
prefered_city
request_status
1
request_feedback
PK id
FK request_id
feedback_date
rating
comments


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]

DS_1 – Oracle (agent_imobiliar) prin DB_LINK

- **Tip model de date**: relațional
- **Format de acces**: SQL
- **Mecanism acces**: DATABASE LINK

- **Descriere mecanism acces:**

  - Se creează un **DB_LINK** din schema FDBO către schema *agent_imobiliar*:

```
CREATE DATABASE LINK agent_imobiliarDB

   CONNECT TO agent_imobiliar IDENTIFIED BY agent_imobiliar

   USING '//localhost:1521/XEPDB1';
```

- **Structuri de acces:**
  - **View-uri locale bazate pe tabele remote:**

- View-ul **cities_VIEW** extrage toate câmpurile relevante despre orașe din tabelul `cities`
- Se află în schema remote `agent_imobiliar`, prin intermediul unui DB_LINK.
- Se folosește pentru a accesa orașele într-un mod local, în schema FDBO.

```
CREATE OR REPLACE VIEW cities_VIEW AS

SELECT id,city_name,state,zip_code
from cities@agent_imobiliarDB;
SELECT * FROM cities_VIEW;
```

- View-ul **owners_view** aduce în schema curentă FDBO date despre proprietari (`owners`) din schema `agent_imobiliar` prin intermediul DB_LINK-ului. Este utilizat pentru a face legătura între proprietăți și proprietarii lor.

```
CREATE OR REPLACE VIEW owners_view AS
SELECT id, full_name, email, phone, created_at
FROM owners@agent_imobiliarDB;
```

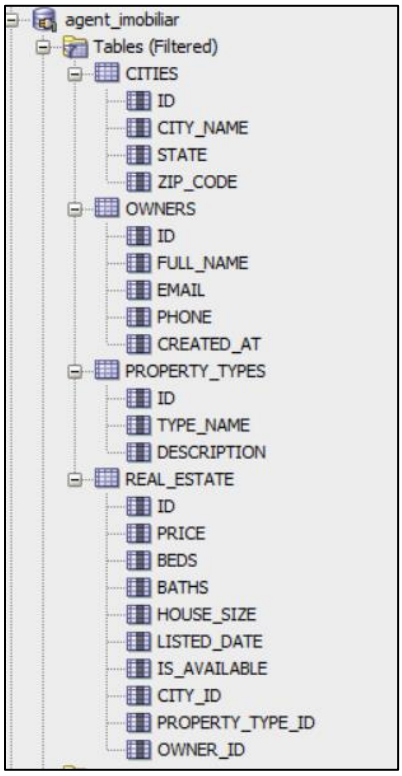
- View-ul **property_types_view** permite accesul la tipurile de proprietăți (`property_types`) din schema `agent_imobiliar`. Fiecare proprietate va fi legată de un tip.

```
CREATE OR REPLACE VIEW property_types_view AS
SELECT id, type_name, description
FROM property_types@agent_imobiliarDB;
```

- View-ul **real_estate_view** extrage toate detaliile despre proprietăți imobiliare din tabelul `real_estate` aflat în schema `agent_imobiliar`. Include referințe la oraș, tipul proprietății și proprietar. Este baza principală pentru view-uri analitice sau de integrare în schema FDBO.

```
CREATE OR REPLACE VIEW real_estate_view AS
SELECT
  id, price, beds, baths, house_size, listed_date, is_available,
  city_id, property_type_id, owner_id
FROM real_estate@agent_imobiliarDB;
```

Tabelele din agent_imobiliar



DS_2 – PostgreSQL (customers, requests, feedback) prin PostgREST

- **Tip model de date:** relațional
- **Format de acces:** JSON (REST)
- **Mecanism acces:** HTTPURITYPE + JSON_TABLE

○ **Descriere mecanism acces:**

- Se folosește **HTTPURITYPE.createuri(...).getclob()** pentru a aduce datele JSON:

```
SELECT HTTPURITYPE.createuri('http://localhost:3000/customers').getclob() as doc
from dual;
```

```
SELECT HTTPURITYPE.createuri('http://localhost:3000/property_requests')
.getclob() AS doc FROM dual)
```

```
SELECT HTTPURITYPE.createuri('http://localhost:3000/request_feedback')
.getclob() AS doc FROM dual)
```

Structuri de acces:

- View-uri Oracle:

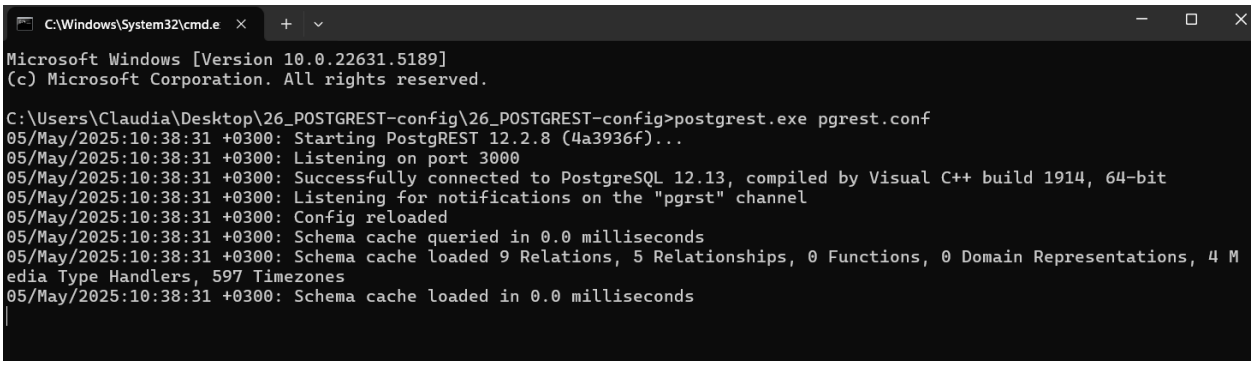
```
CREATE OR REPLACE VIEW customers_view AS
with rest_doc as
    (SELECT HTTPURITYPE.createuri('http://localhost:3000/customers')
    .getclob() as doc from dual)
SELECT
  id, customer_code, full_name, email, phone, registration_date, country, city as customers
FROM  JSON_TABLE( (select doc from rest_doc) , '$[*]'
            COLUMNS (
                id            PATH '$.id'
                , customer_code     PATH '$.customer_code'
                , full_name           PATH '$.full_name'
                , email              PATH '$.email'
                , phone            PATH '$.phone'
                , registration_date PATH '$.registration_date'
                , country PATH '$.country'
                , city PATH '$.city'
                )
);
```

```
CREATE OR REPLACE VIEW property_requests_view AS
WITH rest_doc AS (
  SELECT HTTPURITYPE.createuri('http://localhost:3000/property_requests')
  .getclob() AS doc FROM dual)
SELECT
  request_id, customer_id, request_date, property_type, max_budget, min_surface, preferred_city, request_status
FROM JSON_TABLE(
  (SELECT doc FROM rest_doc), '$[*]'
  COLUMNS (
    request_id     PATH '$.request_id',
    customer_id    PATH '$.customer_id',
    request_date   PATH '$.request_date',
    property_type   PATH '$.property_type',
    max_budget     PATH '$.max_budget',
    min_surface    PATH '$.min_surface',
    preferred_city  PATH '$.preferred_city',
    request_status  PATH '$.request_status'
  )
);
```

```
CREATE OR REPLACE VIEW request_feedback_view AS
WITH rest_doc AS (
  SELECT HTTPURITYPE.createuri('http://localhost:3000/request_feedback')
  .getclob() AS doc FROM dual)
SELECT
  id, request_id, feedback_date, rating, comments
FROM JSON_TABLE(
  (SELECT doc FROM rest_doc), '$[*]'
  COLUMNS (
    id                PATH '$.id',
    request_id    PATH '$.request_id',
    feedback_date  PATH '$.feedback_date',
    rating         PATH '$.rating',
    comments       PATH '$.comments'
  )
);
```

Răspuns JSON în Browser:
URL: http://localhost:3000/customers
- Format răspuns: JSON
- Instrument testat: Browser

Configurare PostgREST:
- PostgREST rulează pe `localhost:3000`



DS_3 – MongoDB (property_history, market_insights etc.) prin RESTHeart

- **Tip model de date**: documentar (NoSQL)
- **Format de acces**: JSON
- **Mecanism acces**: PL/SQL + RESTHeart + JSON_TABLE

- **Descriere mecanism acces:**
  - Funcție PL/SQL personalizată:

  Se creează o funcție care primește:

  - pURL → adresa URL de la care să ia date (ex: http://localhost:8081/property_reviews)
  - pUserPass → string-ul user:password codificat ulterior în Base64

```
CREATE OR REPLACE FUNCTION get_restheart_data_media(pURL VARCHAR2, pUserPass VARCHAR2)
RETURN clob IS
  l_req   UTL_HTTP.req;
  l_resp  UTL_HTTP.resp;
  l_buffer clob;
begin
  l_req  := UTL_HTTP.begin_request(pURL);
  UTL_HTTP.set_header(l_req, 'Authorization', 'Basic ' ||
    UTL_RAW.cast_to_varchar2(UTL_ENCODE.base64_encode(UTL_I18N.string_to_raw(pUserPass, 'AL32UTF8'))));
  l_resp := UTL_HTTP.get_response(l_req);
  UTL_HTTP.READ_TEXT(l_resp, l_buffer);
  UTL_HTTP.end_response(l_resp);
  return l_buffer;
end;
/
```

- Se apelează funcția în view-uri

```
SELECT * FROM JSON_TABLE(
  (SELECT get_restheart_data_media('http://localhost:8081/property_reviews', 'admin:secret') FROM dual),
  '$[*]' COLUMNS (...)
```

Structuri de acces:

- View-uri Oracle:

```
CREATE OR REPLACE VIEW PROPERTY_HISTORY_VIEW_MONGODB AS
WITH json AS (
  SELECT get_restheart_data_media('http://localhost:8081/property_history', 'admin:secret') AS doc
  FROM dual
)
SELECT *
FROM JSON_TABLE(
  (SELECT doc FROM json),
  '$[*]'
  COLUMNS (
    mongo_id VARCHAR2(100) PATH '$."_id"."$oid"',
    property_id      VARCHAR2(100) PATH '$.property_id',
    oracle_id          NUMBER        PATH '$.oracle_id',
    previous_price   NUMBER        PATH '$.previous_price',
    new_price        NUMBER        PATH '$.new_price',
    modification_date VARCHAR2(30) PATH '$.modification_date',
    reason           VARCHAR2(100) PATH '$.reason'
  )
);
```

```
CREATE OR REPLACE VIEW MARKET_INSIGHTS_VIEW_MONGODB AS
WITH json AS (
  SELECT get_restheart_data_media('http://localhost:8081/market_insights', 'admin:secret') AS doc FROM dual
)
SELECT *
FROM JSON_TABLE(
  (SELECT doc FROM json),
  '$[*]'
  COLUMNS (
    mongo_id      VARCHAR2(100) PATH '$."_id"."$oid"',
     oracle_id          NUMBER        PATH '$.oracle_id',
    city          VARCHAR2(100) PATH '$.city',
    average_price NUMBER        PATH '$.average_price',
    average_surface NUMBER      PATH '$.average_surface',
    timestamp     VARCHAR2(30)  PATH '$.timestamp',
    trend         VARCHAR2(50)  PATH '$.trend'
  )
);
```

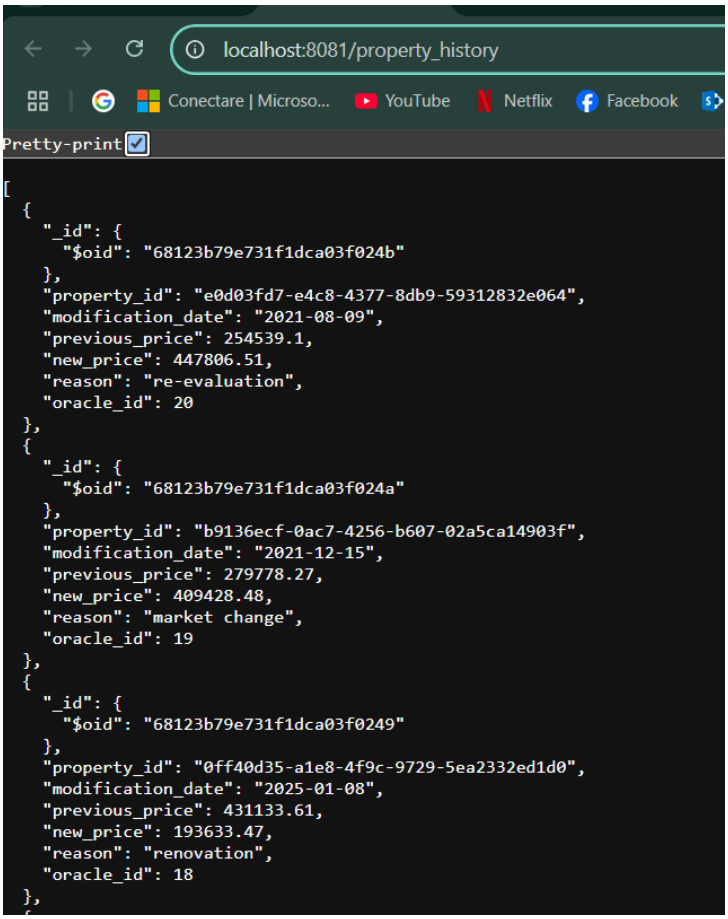
```
CREATE OR REPLACE VIEW PROPERTY_REVIEWS_VIEW_MONGODB AS
WITH json AS (
  SELECT get_restheart_data_media('http://localhost:8081/property_reviews', 'admin:secret') AS doc FROM dual
)
SELECT *
FROM JSON_TABLE(
  (SELECT doc FROM json),
  '$[*]'
  COLUMNS (
    mongo_id       VARCHAR2(100) PATH '$."_id"."$oid"',
    property_id    VARCHAR2(100) PATH '$.property_id',
    oracle_id      NUMBER         PATH '$.oracle_id',
    review_date    VARCHAR2(30)  PATH '$.review_date',
    rating         NUMBER        PATH '$.rating',
    reviewer       VARCHAR2(100) PATH '$.reviewer',
    comment_text   VARCHAR2(400) PATH '$.comment'
  )
);
```

```
CREATE OR REPLACE VIEW SITE_ACTIVITY_LOGS_VIEW_MONGODB AS
WITH json AS (
  SELECT get_restheart_data_media('http://localhost:8081/site_activity_logs', 'admin:secret') AS doc FROM dual
)
SELECT *
FROM JSON_TABLE(
  (SELECT doc FROM json),
  '$[*]'
  COLUMNS (
    mongo_id  VARCHAR2(100) PATH '$."_id"."$oid"',
    user_id   VARCHAR2(100) PATH '$.user_id',
    oracle_id    NUMBER         PATH '$.oracle_id',
    action    VARCHAR2(100) PATH '$.action',
    timestamp VARCHAR2(30)  PATH '$.timestamp',
    ip_address VARCHAR2(50) PATH '$.ip_address'
  )
);
```

Răspuns JSON în Browser:

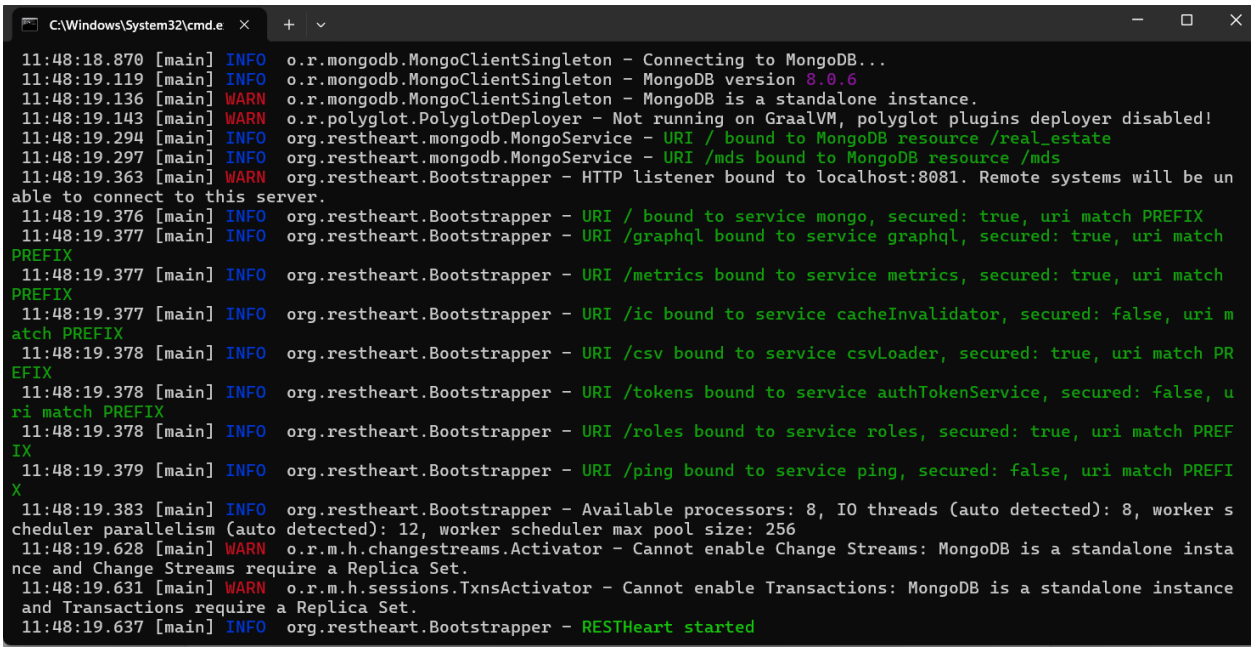
URL: http://localhost:8081/property_history

- Format răspuns: JSON
- Instrument testat: Browser




```
[
  {
    "_id": {
      "$oid": "68123b79e731f1dca03f024b"
    },
    "property_id": "e0d03fd7-e4c8-4377-8db9-59312832e064",
    "modification_date": "2021-08-09",
    "previous_price": 254539.1,
    "new_price": 447806.51,
    "reason": "re-evaluation",
    "oracle_id": 20
  },
  {
    "_id": {
      "$oid": "68123b79e731f1dca03f024a"
    },
    "property_id": "b9136ecf-0ac7-4256-b607-02a5ca14903f",
    "modification_date": "2021-12-15",
    "previous_price": 279778.27,
    "new_price": 409428.48,
    "reason": "market change",
    "oracle_id": 19
  },
  {
    "_id": {
      "$oid": "68123b79e731f1dca03f0249"
    },
    "property_id": "0ff40d35-a1e8-4f9c-9729-5ea2332ed1d0",
    "modification_date": "2025-01-08",
    "previous_price": 431133.61,
    "new_price": 193633.47,
    "reason": "renovation",
    "oracle_id": 18
  },
```

Configurare RESTHeart:
- RESTHeart rulează pe `localhost:8081`



TEMA: L3. Analytical Integration Model: Implementare OLAP views

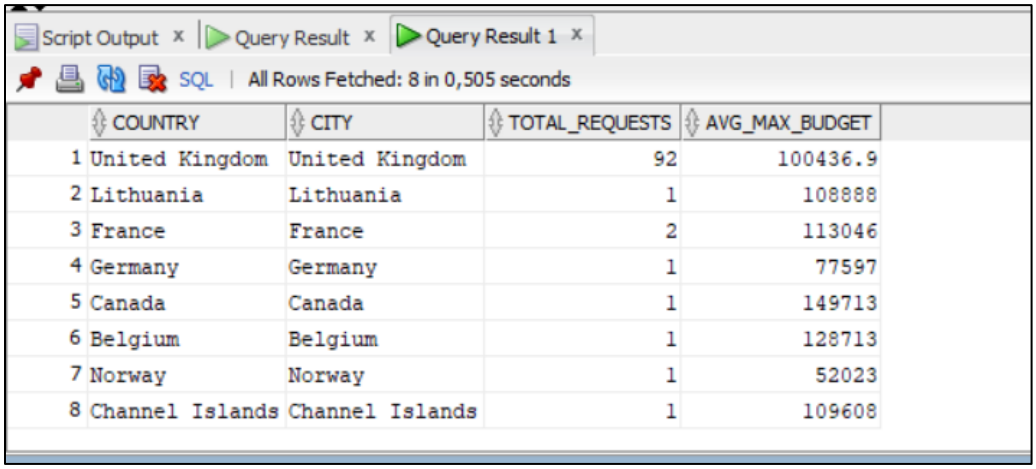
(1) Nivel CONSOLIDARE date

View_Consolidare_1: request_budget_summary_view

- **Descriere**: Rezumat al cererilor imobiliare pe oraș și țară.
- **Surse de date integrate**: DS_1 (PostgreSQL) + DS_2 (Oracle)
- **Definiție SQL:**

```sql
CREATE OR REPLACE VIEW request_budget_summary_view AS
SELECT
    c.country,
    c.customers AS city,
    COUNT(*) AS total_requests,
    ROUND(AVG(TO_NUMBER(pr.max_budget)), 2) AS avg_max_budget
FROM
    property_requests_view pr
JOIN
    customers_view c ON pr.customer_id = c.id
WHERE
    REGEXP_LIKE(pr.max_budget, '^\d+(\.\d+)?$') -- doar bugete numerice valide
GROUP BY
    c.country, c.customers;
```

Rezultat:



Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 8 in 0,505 seconds

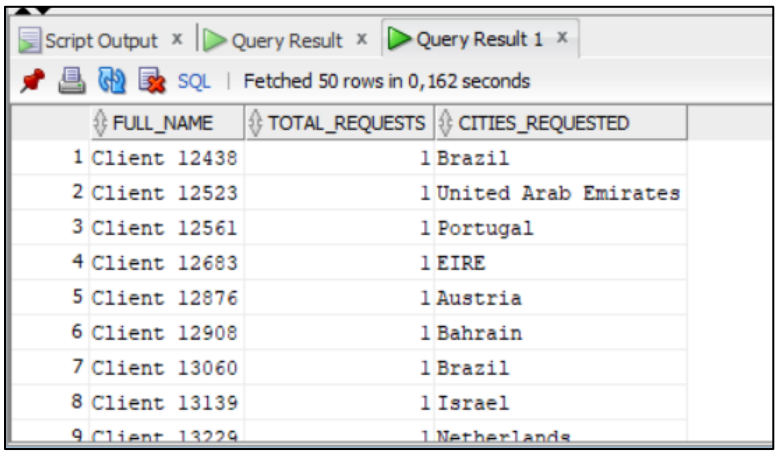
	COUNTRY	CITY	TOTAL_REQUESTS	AVG_MAX_BUDGET
1	United Kingdom	United Kingdom	92	100436.9
2	Lithuania	Lithuania	1	108888
3	France	France	2	113046
4	Germany	Germany	1	77597
5	Canada	Canada	1	149713
6	Belgium	Belgium	1	128713
7	Norway	Norway	1	52023
8	Channel Islands	Channel Islands	1	109608

View_Consolidare_2: customer_request_summary_view

- **Descriere**: Rezumă activitatea fiecărui client și orașele preferate.
- **Surse de date**: DS_1 (PostgreSQL) + DS_2 (Oracle)
- **Definiție SQL**:

```
CREATE OR REPLACE VIEW customer_request_summary_view AS
SELECT
    c.full_name,
    COUNT(*) AS total_requests,
    LISTAGG(DISTINCT pr.preferred_city, ', ') WITHIN GROUP (ORDER BY pr.preferred_city) AS cities_requested
FROM
    customers_view c
JOIN
    property_requests_view pr ON c.id = pr.customer_id
GROUP BY
    c.full_name;
```

Rezultat:



Script Output | Query Result | Query Result 1

SQL | Fetched 50 rows in 0,162 seconds

	FULL_NAME	TOTAL_REQUESTS	CITIES_REQUESTED
1	Client 12438	1	Brazil
2	Client 12523	1	United Arab Emirates
3	Client 12561	1	Portugal
4	Client 12683	1	EIRE
5	Client 12876	1	Austria
6	Client 12908	1	Bahrain
7	Client 13060	1	Brazil
8	Client 13139	1	Israel
9	Client 13229	1	Netherlands

View_Consolidare_3: latest_requests_by_customer_view

- **Descriere**: Ultima cerere imobiliară pentru fiecare client.
- **Surse de date**: DS_1 (PostgreSQL) + DS_2(Oracle)
- **Definiție SQL**:

```sql
CREATE OR REPLACE VIEW latest_requests_by_customer_view AS
SELECT
    c.full_name,
    pr.request_date,
    pr.preferred_city,
    pr.property_type,
    pr.max_budget
FROM
    customers_view c
JOIN
    property_requests_view pr ON c.id = pr.customer_id
WHERE
    pr.request_date = (
        SELECT MAX(pr2.request_date)
        FROM property_requests_view pr2
        WHERE pr2.customer_id = pr.customer_id
    );
```

Rezultat:

Script Output | Query Result | Query Result 1

SQL | Fetched 50 rows in 0,273 seconds

	FULL_NAME	REQUEST_DATE	PREFERRED_CITY	PROPERTY_TYPE	MAX_BUDGET
1	Client 15716	2025-02-19	Finland	Apartment	133806
2	Client 17734	2025-01-02	RSA	Studio	99857
3	Client 14951	2025-01-18	Australia	Apartment	58621
4	Client 13890	2025-01-16	EIRE	Apartment	100755
5	Client 17839	2025-02-07	Bahrain	Studio	130632
6	Client 14495	2025-02-26	United Arab Emirates	Apartment	149856
7	Client 13593	2025-01-14	United Kingdom	Studio	112988
8	Client 14484	2025-03-30	Switzerland	Studio	72150
9	Client 14036	2025-01-17	Belgium	Studio	44530
10	Client 17806	2025-02-28	Lebanon	Studio	82465
11	Client 17924	2025-02-04	Unspecified	Studio	132286
12	Client 16638	2025-03-16	Channel Islands	Studio	106398
13	Client 14359	2025-02-12	Lithuania	Apartment	121119
14	Client 17289	2025-01-03	Canada	Apartment	140969

View_Consolidare_4: property_price_changes_view

- **Descriere**: Modificările de preț pentru fiecare proprietate.
- **Surse de date**: DS_2 (Oracle) + DS_3 (MongoDB)
- **Definiție SQL**:

```
CREATE OR REPLACE VIEW property_price_changes_view AS
SELECT
    r.id AS property_id,
    r.price AS current_price,
    ph.previous_price,
    ph.new_price,
    ph.modification_date,
    ph.reason
FROM
    real_estate_view r
JOIN
    PROPERTY_HISTORY_VIEW_MONGODB ph
    ON ph.oracle_id = r.id;
```

Rezultat:

	PROPERTY_ID	CURRENT_PRICE	PREVIOUS_PRICE	NEW_PRICE	MODIFICATION_DATE	REASON
1	9	350000	412691.19	305163.33	2023-08-27	owner request
2	8	500000	243369.76	323123.71	2023-12-07	owner request
3	7	350000	285183.21	180870.14	2020-03-10	re-evaluation
4	6	450000	477547.33	256372.79	2022-04-19	re-evaluation
5	5	600000	261814.78	310413.49	2020-08-07	market change
6	4	500000	138672.39	367153.13	2023-05-13	re-evaluation
7	3	250000	358195.11	389940	2023-04-11	market change
8	2	400000	483628.94	266956.77	2020-01-14	renovation

View_Consolidare_5: property_price_evolution_view

- **Descriere**: Evoluția prețurilor imobiliare.
- **Surse de date**: DS_2 + DS_3
- **Definiție SQL**:

```sql
CREATE OR REPLACE VIEW property_price_evolution_view AS
SELECT
    r.id AS property_id,
    r.price AS current_price,
    r.city_id,
    ph.previous_price,
    ph.new_price,
    ph.modification_date,
    ph.reason
FROM
    real_estate_view r
JOIN
    PROPERTY_HISTORY_VIEW_MONGODB ph
    ON r.id = ph.oracle_id
ORDER BY
    r.id, ph.modification_date;
```

Rezultat:

	PROPERTY_ID	CURRENT_PRICE	CITY_ID	PREVIOUS_PRICE	NEW_PRICE	MODIFICATION_DATE	REASON
1	1	350000	1	221986.17	280569.03	2020-12-22	market change
2	2	400000	2	483628.94	266956.77	2020-01-14	renovation
3	3	250000	3	358195.11	389940	2023-04-11	market change
4	4	500000	4	138672.39	367153.13	2023-05-13	re-evaluation
5	5	600000	5	261814.78	310413.49	2020-08-07	market change
6	6	450000	6	477547.33	256372.79	2022-04-19	re-evaluation
7	7	350000	7	285183.21	180870.14	2020-03-10	re-evaluation
8	8	500000	8	243369.76	323123.71	2023-12-07	owner request
9	9	350000	9	412691.19	305163.33	2023-08-27	owner request

View_Consolidare_6: property_review_analysis_view

- **Descriere**: Analiza recenziilor imobiliare.
- **Surse de date**: DS_2 + DS_3
- **Definiție SQL**:

```
CREATE OR REPLACE VIEW property_review_analysis_view AS
SELECT
    r.id AS property_id,
    r.price,
    r.city_id,
    pr.rating,
    pr.review_date,
    pr.reviewer,
    pr.comment_text
FROM
    real_estate_view r
JOIN
    PROPERTY_REVIEWS_VIEW_MONGODB pr ON pr.oracle_id = r.id;
```

Rezultat:

	PROPERTY_ID	PRICE	CITY_ID	RATING	REVIEW_DATE	REVIEWER	COMMENT_TEXT
1	9	350000	9	1	2025-03-21	Philip Wagner	Either create morning on find medical tend buy worry.
2	8	500000	8	4	2025-01-30	Kayla Huang	Party score determine result today impact finish.
3	7	350000	7	3	2025-02-07	Mrs. Sharon Fields MD	Others ability not end day size.
4	6	450000	6	3	2025-03-08	Ronald Davis	Time test true time expect visit go just.
5	5	600000	5	1	2025-04-03	Chase Peters	Coach appear hundred entire language figure.
6	4	500000	4	3	2025-04-25	Matthew Rose	As before your speak we report recently third man pick.
7	3	250000	3	3	2025-03-27	David Roach	Cause character call face both reveal.
8	2	400000	2	1	2025-01-02	Alicia Valdez	Indicate trade American war player the Republican thank down rule risk order bring.
9	1	350000	1	2	2025-04-20	Charlene Rivera	Would fall bed Republican own middle total beautiful program especially the box.

View_Consolidare_7: site_activity_monitoring_view

- **Descriere**: Monitorizarea acțiunilor utilizatorilor.
- **Surse de date**: DS_1 + DS_3
- **Definiție SQL**:

```
CREATE OR REPLACE VIEW site_activity_monitoring_view AS
SELECT
    c.full_name,
    sa.action,
    sa.timestamp,
    sa.ip_address
FROM
    customers_view c
JOIN
    SITE_ACTIVITY_LOGS_VIEW_MONGODB sa ON sa.oracle_id = c.id;
```

Rezultat:

	FULL_NAME	ACTION	TIMESTAMP	IP_ADDRESS
1	Client 13747	request_info	2025-04-28T07:46:13	136.166.139.232
2	Client 13705	logout	2025-03-19T11:58:38	132.173.245.211
3	Client 17548	logout	2025-04-06T14:05:38	149.220.225.72
4	Client 17511	view_property	2025-01-20T09:35:48	6.225.170.216
5	Client 12431	view_property	2025-04-21T02:05:12	30.0.24.95
6	Client 16250	request_info	2025-03-06T08:38:43	199.79.231.33
7	Client 16029	request_info	2025-01-02T02:43:09	218.190.146.105
8	Client 17420	logout	2025-02-05T14:47:33	185.4.21.176
9	Client 18074	view_property	2025-01-31T01:48:17	164.51.36.85
10	Client 16098	request_info	2025-03-03T16:10:09	37.200.50.126
11	Client 14527	login	2025-02-01T05:44:07	113.224.106.109
12	Client 15311	login	2025-04-10T02:17:56	223.237.246.84
13	Client 17809	edit_profile	2025-02-17T21:16:44	214.112.182.143
14	Client 14688	request_info	2025-03-11T00:43:29	135.4.103.27

(2) Schema analitică ROLAP

Tabela_de_fapte_1: property_price_fact

- **Descriere**: Integrează date cantitative despre modificările de preț ale proprietăților
- **Surse de date integrate**: Oracle + MongoDB
- **Definiție**: fraza DDL SQL

```
CREATE OR REPLACE VIEW property_price_fact AS
SELECT
    r.id AS property_id,
    r.price AS current_price,
    ph.previous_price,
    ph.new_price,
    TO_DATE(ph.modification_date, 'YYYY-MM-DD') AS modification_date,
    r.city_id
FROM
    real_estate_view r
JOIN
    PROPERTY_HISTORY_VIEW_MONGODB ph ON ph.oracle_id = r.id;
```

Rezultat:

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 9 in 0,119 seconds

	PROPERTY_ID	CURRENT_PRICE	PREVIOUS_PRICE	NEW_PRICE	MODIFICATION_DATE	CITY_ID
1	9	350000	412691.19	305163.33	27-AUG-23	9
2	8	500000	243369.76	323123.71	07-DEC-23	8
3	7	350000	285183.21	180870.14	10-MAR-20	7
4	6	450000	477547.33	256372.79	19-APR-22	6
5	5	600000	261814.78	310413.49	07-AUG-20	5
6	4	500000	138672.39	367153.13	13-MAY-23	4
7	3	250000	358195.11	389940	11-APR-23	3
8	2	400000	483628.94	266956.77	14-JAN-20	2

Tabela_de_fapte_2: property_requests_fact

- **Descriere:** Integrează cererile și feedbackul clienților
- **Surse de date integrate:** PostgreSQL + MongoDB
- **Definiție:** fraza DDL SQL

```
CREATE OR REPLACE VIEW property_requests_fact AS
SELECT
    pr.request_id,
    pr.customer_id,
    pr.max_budget,
    pr.min_surface,
    f.rating
FROM
    property_requests_view pr
LEFT JOIN request_feedback_view f ON f.request_id = pr.request_id;
```

Rezultat:

Script Output | Query Result | Query Result 1

SQL | Fetched 100 rows in 0,147 seconds

	REQUEST_ID	CUSTOMER_ID	MAX_BUDGET	MIN_SURFACE	RATING
7	7	149	95368	68	5
8	8	56	103620	63	4
9	9	2111	141927	50	3
10	10	274	129024	55	1
11	11	2862	140969	32	3
12	12	1433	130868	65	3
13	13	2637	109463	119	5
14	14	1734	43675	40	2

View dimensional: dim_properties_extended

- **Descriere:** Dimensiune extinsă pentru proprietăți, care combină informații despre proprietate, oraș și tipul de proprietate.
- **Surse de date integrate:** DS_2: Oracle
- **Definiție:** fraza DDL SQL

```sql
CREATE OR REPLACE VIEW dim_properties_extended AS
SELECT
    r.id AS property_id,
    r.price,
    r.beds,
    r.baths,
    r.house_size,
    r.listed_date,
    r.is_available,
    c.city_name,
    c.state,
    pt.type_name
FROM
    real_estate_view r
JOIN
    cities_view c ON r.city_id = c.id
JOIN
    property_types_view pt ON r.property_type_id = pt.id;
```

Rezultat:

Script Output × | Query Result × | Query Result 1 ×

SQL | All Rows Fetched: 9 in 0,063 seconds

	PROPERTY_ID	PRICE	BEDS	BATHS	HOUSE_SIZE	LISTED_DATE	IS_AVAILABLE	CITY_NAME	STATE	TYPE_NAME
1	1	350000	4	3	2500	01-OCT-23	Y	Austin	TX	House
2	3	250000	3	2	1800	15-SEP-23	N	Chicago	IL	Condo
3	7	350000	3	2	2500	15-OCT-23	Y	Dallas	TX	Apartment
4	6	450000	4	3	3500	10-OCT-23	Y	Houston	TX	House
5	5	600000	7	6	6000	01-DEC-23	N	Los Angeles	CA	Studio
6	2	400000	5	4	3500	02-OCT-23	Y	Miami	FL	Apartment
7	4	500000	6	5	4500	05-NOV-23	Y	New York	NY	Villa
8	8	500000	5	4	4000	10-NOV-23	N	Phoenix	AZ	Condo

View dimensional: dim_customers_enriched

- **Descriere:** Dimensiune detaliată a clienților, cu informații personale și localizare.
- **Surse de date integrate:** DS_1: PostgreSQL (prin customers_view REST)
- **Definiție:** fraza DDL SQL

```sql
CREATE OR REPLACE VIEW dim_customers_enriched AS
SELECT
    id AS customer_id,
    customer_code,
    full_name,
    email,
    phone,
    registration_date,
    country,
    customers AS city
FROM
    customers_view;
```

Rezultat:

Script Output × | Query Result × | Query Result 1 ×

SQL | Fetched 50 rows in 0,083 seconds

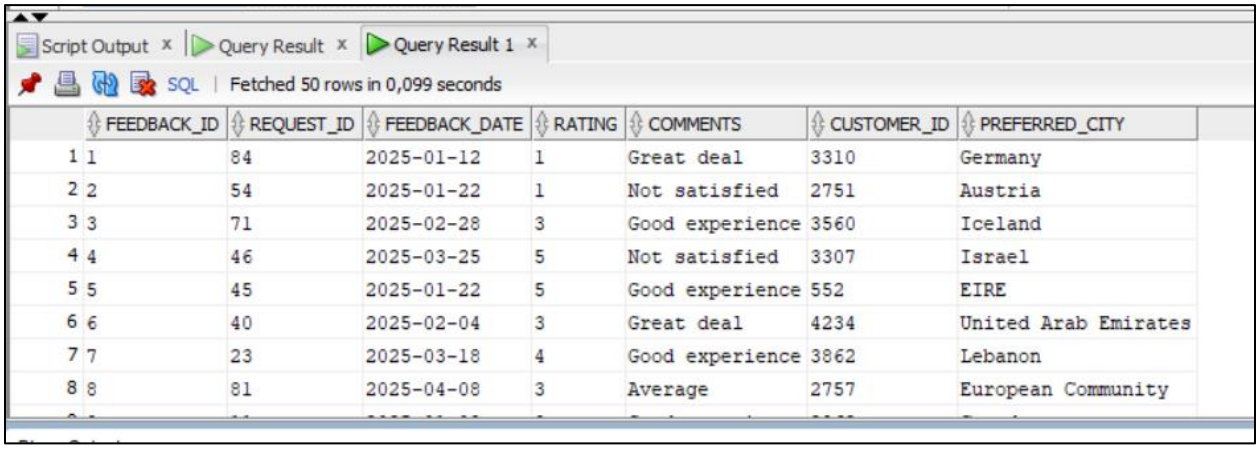
	CUSTOMER_ID	CUSTOMER_CODE	FULL_NAME	EMAIL	PHONE	REGISTRATION_DATE	COUNTRY	CITY
1	1	C17850.0	Client 17850	17850@client.com	4070017850	2022-02-23	United Kingdom	United Kingdom
2	2	C13047.0	Client 13047	13047@client.com	4070013047	2022-12-25	United Kingdom	United Kingdom
3	3	C12583.0	Client 12583	12583@client.com	4070012583	2022-11-30	France	France
4	4	C13748.0	Client 13748	13748@client.com	4070013748	2022-01-03	United Kingdom	United Kingdom
5	5	C15100.0	Client 15100	15100@client.com	4070015100	2022-08-10	United Kingdom	United Kingdom
6	6	C15291.0	Client 15291	15291@client.com	4070015291	2022-01-25	United Kingdom	United Kingdom
7	7	C14688.0	Client 14688	14688@client.com	4070014688	2022-09-21	United Kingdom	United Kingdom
8	8	C17809.0	Client 17809	17809@client.com	4070017809	2022-04-03	United Kingdom	United Kingdom

View dimensional: dim_feedback_rating

- **Descriere:** Dimensiune pentru evaluările date cererilor imobiliare.
- **Surse de date integrate:**
  DS_1 + DS_3 (PostgreSQL + MongoDB, prin feedback)
- **Definiție:** fraza DDL SQL

```sql
CREATE OR REPLACE VIEW dim_feedback_rating AS
SELECT
    rf.id AS feedback_id,
    rf.request_id,
    rf.feedback_date,
    rf.rating,
    rf.comments,
    pr.customer_id,
    pr.preferred_city
FROM
    request_feedback_view rf
JOIN
    property_requests_view pr ON rf.request_id = pr.request_id;
```

Rezultat:



Script Output | Query Result | Query Result 1

SQL | Fetched 50 rows in 0,099 seconds

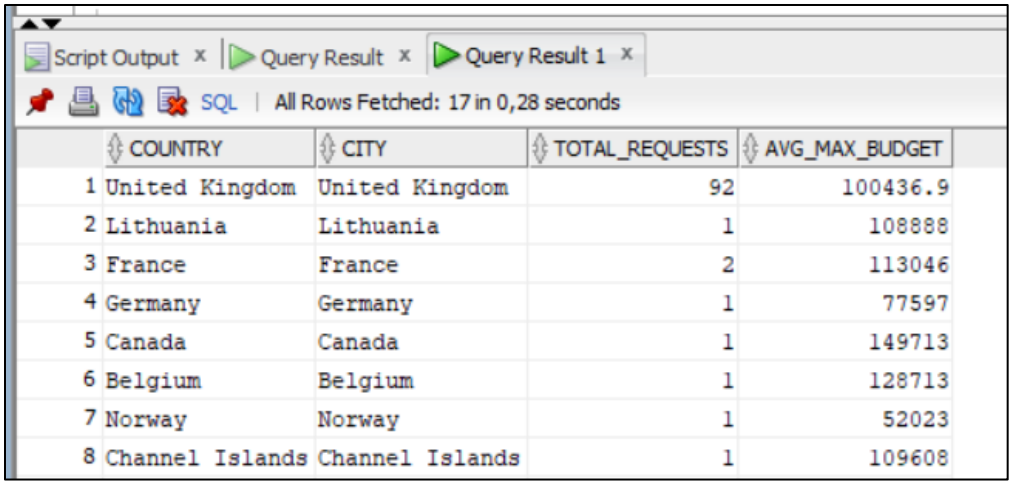
	FEEDBACK_ID	REQUEST_ID	FEEDBACK_DATE	RATING	COMMENTS	CUSTOMER_ID	PREFERRED_CITY
1	1	84	2025-01-12	1	Great deal	3310	Germany
2	2	54	2025-01-22	1	Not satisfied	2751	Austria
3	3	71	2025-02-28	3	Good experience	3560	Iceland
4	4	46	2025-03-25	5	Not satisfied	3307	Israel
5	5	45	2025-01-22	5	Good experience	552	EIRE
6	6	40	2025-02-04	3	Great deal	4234	United Arab Emirates
7	7	23	2025-03-18	4	Good experience	3862	Lebanon
8	8	81	2025-04-08	3	Average	2757	European Community

View_Analitic_OLAP_1: olap_request_summary_view

- **Descriere**: Analiză agregată a cererilor imobiliare pe oraș și țară
- **Surse de date integrate**: PostgreSQL + Oracle
- **Tip procesare analitică:** Clauză **ROLLUP** + funcții agregare COUNT, AVG
- **Definiție:** fraza DDL SQL

```sql
CREATE OR REPLACE VIEW olap_request_summary_view AS
SELECT
    c.country,
    c.customers AS city,
    COUNT(*) AS total_requests,
    ROUND(AVG(TO_NUMBER(pr.max_budget)), 2) AS avg_max_budget
FROM
    customers_view c
JOIN
    property_requests_view pr ON c.id = pr.customer_id
WHERE
    REGEXP_LIKE(pr.max_budget, '^\d+(\.\d+)?$') -- doar bugete valide numeric
GROUP BY
    ROLLUP (c.country, c.customers);
```

Rezultat:



Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 17 in 0,28 seconds

	COUNTRY	CITY	TOTAL_REQUESTS	AVG_MAX_BUDGET
1	United Kingdom	United Kingdom	92	100436.9
2	Lithuania	Lithuania	1	108888
3	France	France	2	113046
4	Germany	Germany	1	77597
5	Canada	Canada	1	149713
6	Belgium	Belgium	1	128713
7	Norway	Norway	1	52023
8	Channel Islands	Channel Islands	1	109608

View_Analitic_OLAP_2: olap_market_trend_cube_view

- **Descriere**: Agregare CUBE a tendințelor pieței imobiliare pe oraș și trend
- **Surse de date integrate**: Oracle + MongoDB
- **Tip procesare analitică:** Clauză **CUBE** + funcții agregare AVG, COUNT
- **Definiție:** fraza DDL SQL

```
CREATE OR REPLACE VIEW olap_market_trend_cube_view AS
SELECT
    c.city_name,
    mi.trend,
    ROUND(AVG(mi.average_price), 2) AS avg_price,
    ROUND(AVG(mi.average_surface), 2) AS avg_surface,
    COUNT(*) AS total_entries
FROM
    cities_view c
JOIN
    MARKET_INSIGHTS_VIEW_MONGODB mi
    ON LOWER(c.city_name) = LOWER(mi.city)
GROUP BY
    CUBE (c.city_name, mi.trend);
```

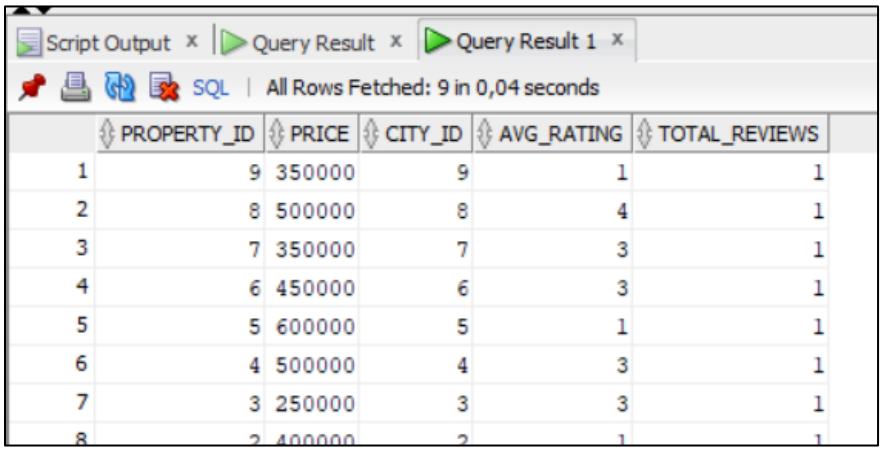
Rezultat:

View_Analitic_OLAP_3: olap_review_summary_view

- **Descriere**: Rating mediu și număr total recenzii per proprietate
- **Surse de date integrate**: Oracle + MongoDB
- **Tip procesare analitică:** Funcții de agregare AVG, COUNT, GROUP BY
- **Definiție:** fraza DDL SQL

```sql
CREATE OR REPLACE VIEW olap_review_summary_view AS
SELECT
    r.id AS property_id,
    r.price,
    r.city_id,
    ROUND(AVG(pr.rating), 2) AS avg_rating,
    COUNT(*) AS total_reviews
FROM
    real_estate_view r
JOIN
    PROPERTY_REVIEWS_VIEW_MONGODB pr ON pr.oracle_id = r.id
GROUP BY
    r.id, r.price, r.city_id;
```

Rezultat:



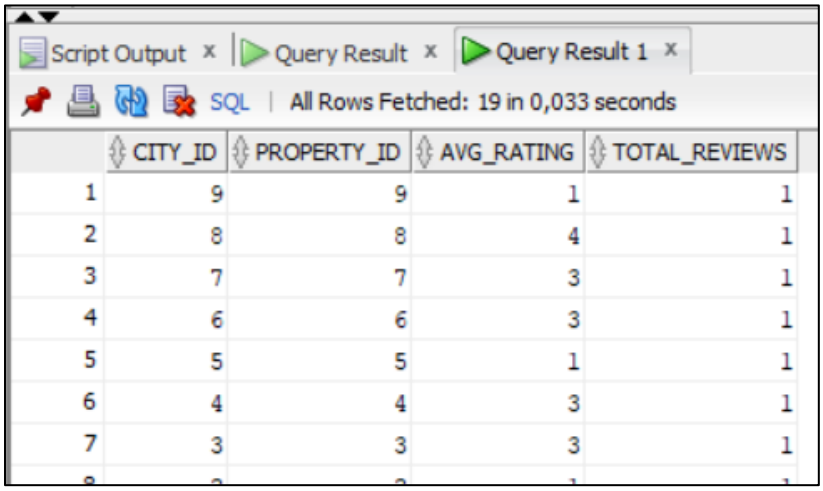
	PROPERTY_ID	PRICE	CITY_ID	AVG_RATING	TOTAL_REVIEWS
1	9	350000	9	1	1
2	8	500000	8	4	1
3	7	350000	7	3	1
4	6	450000	6	3	1
5	5	600000	5	1	1
6	4	500000	4	3	1
7	3	250000	3	3	1
8	2	400000	2	1	1

View_Analitic_OLAP_4: olap_review_summary_rollup_view

- **Descriere**: Agregare ROLLUP a ratingurilor și recenziilor pe oraș și proprietate
- **Surse de date integrate**: Oracle + MongoDB
- **Tip procesare analitică:** Clauză **ROLLUP** + funcții agregare AVG, COUNT
- **Definiție**: fraza DDL SQL

```sql
CREATE OR REPLACE VIEW olap_review_summary_rollup_view AS
SELECT
    r.city_id,
    r.id AS property_id,
    ROUND(AVG(pr.rating), 2) AS avg_rating,
    COUNT(*) AS total_reviews
FROM
    real_estate_view r
JOIN
    PROPERTY_REVIEWS_VIEW_MONGODB pr
    ON pr.oracle_id = r.id
GROUP BY
    ROLLUP (r.city_id, r.id);
```

Rezultat:



	CITY_ID	PROPERTY_ID	AVG_RATING	TOTAL_REVIEWS
1	9	9	1	1
2	8	8	4	1
3	7	7	3	1
4	6	6	3	1
5	5	5	1	1
6	4	4	3	1
7	3	3	3	1

View_Analitic_OLAP_4: olap_monthly_city_reviews_view

- **Descriere**: Analiza lunară a ratingurilor acordate în recenzii, agregate pe orașe
- **Tip procesare analitică:** funcții de agregare (AVG, COUNT) + extragere lună cu TO_CHAR
- **Surse de date integrate**: Oracle + MongoDB
- **Definiție**: fraza DDL SQL

```sql
CREATE OR REPLACE VIEW olap_monthly_city_reviews_view AS
SELECT
    c.city_name,
    TO_CHAR(TO_DATE(pr.review_date, 'YYYY-MM-DD'), 'YYYY-MM') AS review_month,
    ROUND(AVG(pr.rating), 2) AS avg_rating,
    COUNT(*) AS total_reviews
FROM
    real_estate_view r
JOIN
    PROPERTY_REVIEWS_VIEW_MONGODB pr ON pr.oracle_id = r.id
JOIN
    cities_view c ON r.city_id = c.id
GROUP BY
    ROLLUP (c.city_name, TO_CHAR(TO_DATE(pr.review_date, 'YYYY-MM-DD'), 'YYYY-MM'))
ORDER BY
    c.city_name, review_month;
```

Rezultat:

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 19 in 0,261 seconds

	CITY_NAME	REVIEW_MONTH	AVG_RATING	TOTAL_REVIEWS
1	Austin	2025-04	2	1
2	Austin	(null)	2	1
3	Chicago	2025-03	3	1
4	Chicago	(null)	3	1
5	Dallas	2025-02	3	1
6	Dallas	(null)	3	1
7	Houston	2025-03	3	1
8	Houston	(null)	3	1

View_Analitic_OLAP_5: olap_feedback_by_city_view

- **Descriere**: Analiză a scorurilor medii de feedback pentru cererile imobiliare, grupate pe oraș
- **Tip procesare analitică**: funcții de agregare (AVG, COUNT) + ROLLUP
- **Surse de date integrate**: PostgreSQL
- **Definiție**: fraza DDL SQL

```sql
CREATE OR REPLACE VIEW olap_feedback_by_city_view AS
SELECT
    pr.preferred_city,
    ROUND(AVG(rf.rating), 2) AS avg_rating,
    COUNT(*) AS total_feedbacks
FROM
    property_requests_view pr
JOIN
    request_feedback_view rf
    ON pr.request_id = rf.request_id
GROUP BY
    ROLLUP (pr.preferred_city);
```

Rezultat:

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 34 in 0,139 seconds

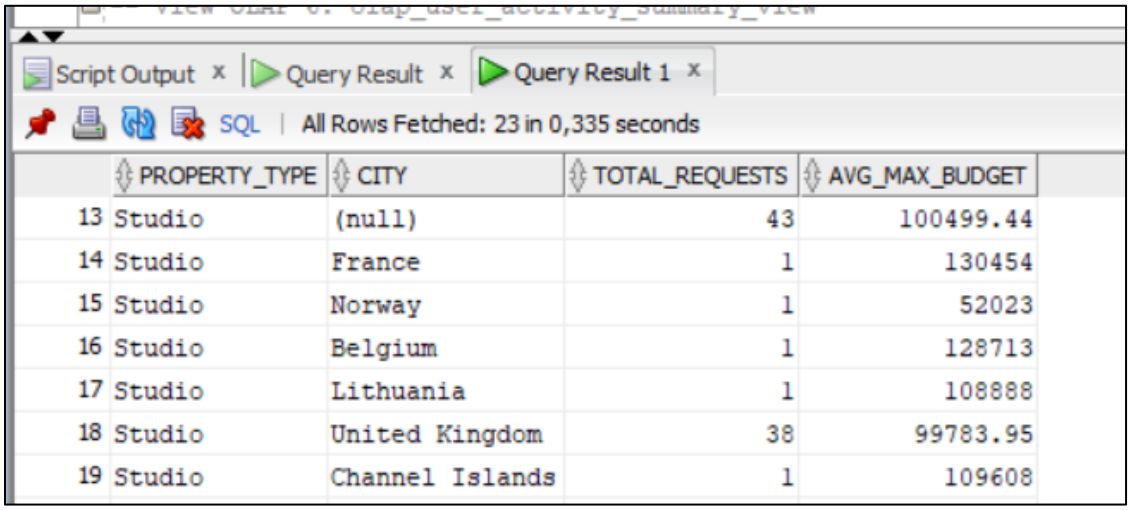
	PREFERRED_CITY	AVG_RATING	TOTAL_FEEDBACKS
1	Germany	1.67	3
2	Austria	1.5	2
3	Iceland	3	3
4	Israel	4	2
5	EIRE	3.67	3
6	United Arab Emirates	3.33	3
7	Lebanon	3	3
8	European Community	3	2

View_Analitic_OLAP_6: olap_property_size_price_cube_view

- **Descriere**: Analiză multidimensională a proprietăților listate, în funcție de mărime și preț
- **Tip procesare analitică**: funcții de agregare (AVG, COUNT) + operator CUBE
- **Surse de date integrate**: Oracle
- **Definiție**: fraza DDL SQL

```sql
CREATE OR REPLACE VIEW olap_property_type_city_cube_view AS
SELECT
    pr.property_type,
    c.customers AS city,
    COUNT(*) AS total_requests,
    ROUND(AVG(TO_NUMBER(pr.max_budget)), 2) AS avg_max_budget
FROM
    property_requests_view pr
JOIN
    customers_view c ON pr.customer_id = c.id
WHERE
    REGEXP_LIKE(pr.max_budget, '^\d+(\.\d+)?$')
GROUP BY
    CUBE (pr.property_type, c.customers);
```

Rezultat:



Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 23 in 0,335 seconds

	PROPERTY_TYPE	CITY	TOTAL_REQUESTS	AVG_MAX_BUDGET
13	Studio	(null)	43	100499.44
14	Studio	France	1	130454
15	Studio	Norway	1	52023
16	Studio	Belgium	1	128713
17	Studio	Lithuania	1	108888
18	Studio	United Kingdom	38	99783.95
19	Studio	Channel Islands	1	109608

View_Analitic_OLAP_7: olap_feedback_by_city_view

- **Descriere**: Analiză a scorurilor medii de feedback pentru cererile imobiliare, grupate pe oraș
- **Tip procesare analitică**: funcții de agregare (AVG, COUNT) + ROLLUP
- **Surse de date integrate**: PostgreSQL
- **Definiție**: fraza DDL SQL

```sql
CREATE OR REPLACE VIEW olap_feedback_by_city_view AS
SELECT
    pr.preferred_city,
    ROUND(AVG(rf.rating), 2) AS avg_rating,
    COUNT(*) AS total_feedbacks
FROM
    property_requests_view pr
JOIN
    request_feedback_view rf
    ON pr.request_id = rf.request_id
GROUP BY
    ROLLUP (pr.preferred_city);
```

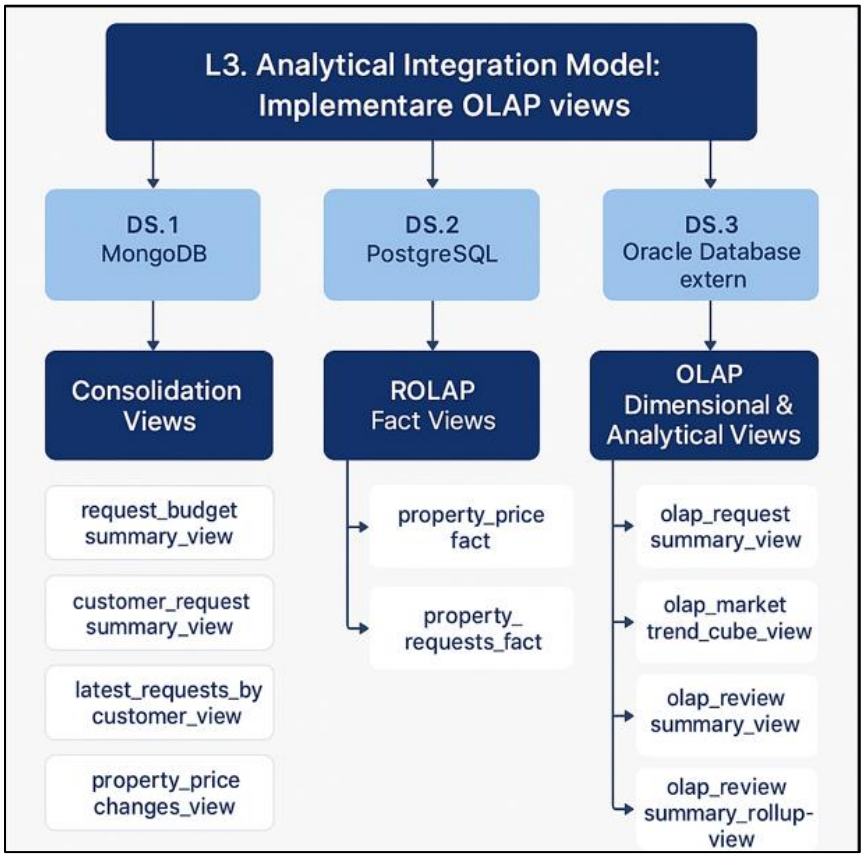
Rezultat:

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 34 in 0,494 seconds

	PREFERRED_CITY	AVG_RATING	TOTAL_FEEDBACKS
1	Germany	1.67	3
2	Austria	1.5	2
3	Iceland	3	3
4	Israel	4	2
5	EIRE	3.67	3
6	United Arab Emirates	3.33	3
7	Lebanon	3	3
8	European Community	3	2

DIAGRAMA



Implementare ORDS and APEX Views

1) Nivel/Model RESTful – acces la view-uri prin ORDS (în JSON/XML)

View OLAP (olap_monthly_city_reviews_view):

- **View_Resursa_REST_1: REST_OLAP_REVIEWS**
  - **Tabela/view țintă**: olap_monthly_city_reviews_view
  - **Detalii de implementare**:
    - Acces REST configurat prin ORDS
    - View deja creat în Oracle SQL Developer
  - **Fraze SQL**:

```
CREATE OR REPLACE VIEW olap_monthly_city_reviews_view AS
SELECT
    c.city_name,
    TO_CHAR(TO_DATE(pr.review_date, 'YYYY-MM-DD'), 'YYYY-MM') AS review_month,
    ROUND(AVG(pr.rating), 2) AS avg_rating,
    COUNT(*) AS total_reviews
FROM
    real_estate_view r
JOIN
    PROPERTY_REVIEWS_VIEW_MONGODB pr ON pr.oracle_id = r.id
JOIN
    cities_view c ON r.city_id = c.id
GROUP BY
    ROLLUP (c.city_name, TO_CHAR(TO_DATE(pr.review_date, 'YYYY-MM-DD'), 'YYYY-MM'))
ORDER BY
    c.city_name, review_month;
```

  - **PL/SQL sau cod implicat**:

```
CREATE OR REPLACE FUNCTION get_olap_reviews RETURN CLOB IS
BEGIN
  RETURN (SELECT JSON_OBJECT(*) FROM olap_monthly_city_reviews_view);
END;
```

  - **URL resursă RESTful**:
    http://localhost:8080/ords/fdbo/olap_reviews/
  - **Descriere concretă**:
    Returnează media ratingurilor și nr. de recenzii lunar, pe oraș, în format JSON

- **Dovadă vizuală**:
  Include un printscreen din Postman sau browser cu rezultatul JSON sau XML

2) Nivel/Model UIX – Aplicație APEX

Pagina: olap_monthly_city_reviews

- **View** : olap_monthly_city_reviews_view
- **Detalii de implementare**:
  - Region tip Chart → Bar Chart
  - Sursa: Local Database, SQL Query:

```
SELECT city_name, review_month, avg_rating, total_reviews
FROM olap_monthly_city_reviews_view;
```

- **Specificații sumare**:
  - Label = city_name
  - Value = avg_rating
  - Series Name = review_month (dacă faci stacked chart)
- **URL pagină**:
  http://localhost:8080/ords/olap_monthly_city_reviews/
- **Descriere grafică**
- Chart tip Bar pentru olap_monthly_city_reviews