

Universitatea din București  
Facultatea de Matematică și Informatică  
Informatică ID

Sparkle – Proiect Baze de Date  
Stratulat Vladimir

Coordonator: Lect. Dr. Iulia Banu Demergian

## CUPRINS

Descrierea modelului real, a utilității acestuia și a regulilor de funcționare. ....	4
Prezentarea constrângerilor (restricții, reguli) impuse asupra modelului. ....	4
Descrierea entităților, incluzând precizarea cheii primare. ....	4
❖ Descrierea relațiilor, incluzând precizarea cardinalității acestora. ....	5
❖ Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor. ....	5
❖ Realizarea diagramei entitate-relație corespunzătoare descrierii de la punctele 3-5. ....	8
❖ Realizarea diagramei conceptuale corespunzătoare diagramei entitate-relație proiectate la punctul 6. Diagrama conceptuală trebuie să conțină minimum 6 tabele (fără considerarea subentităților), dintre care cel puțin un tabel asociativ.....	8
❖ Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectate la punctul 7. ....	9
❖ Realizarea normalizării până la forma normală 3(FN1-FN3).....	9
❖ Dovezi rulare cod într-o baza de date ORACLE .....	10
❖ Dovada a faptului că datele sunt inserate în tabel: .....	13
➤ Users .....	13
➤ Products .....	13
➤ User_Address.....	13
➤ User_payment .....	14
➤ Countries.....	14
➤ Regions .....	14
➤ Reviews.....	15
➤ Orders.....	15
➤ Product_from_order.....	16
➤ Publishers.....	16
➤ Genres .....	17
➤ Product_genres.....	17
➤ Product_type .....	18
❖ Formulați în limbaj natural și implementați 5 cereri SQL complexe ce vor utiliza, în ansamblul lor, următoarele elemente:.....	18
➤ Subcereri sincronizate în care intervin cel puțin 3 tabele: .....	18
➤ Subcereri nesincronizate în clauza FROM.....	19
➤ Grupări de date cu subcereri nesincronizate în care intervin cel puțin 3 tabele, funcții grup, filtrare la nivel de grupuri (în cadrul aceleiași cereri) .....	20
➤ Ordonări și utilizarea funcțiilor NVL și DECODE (în cadrul aceleiași cereri) .....	21
➤ Utilizarea a cel puțin 2 funcții pe șiruri de caractere, 2 funcții pe date calendaristice, a cel puțin unei expresii CASE.....	22
❖ Implementarea a 3 operații de actualizare și de ștergere a datelor utilizând subcereri .....	24

❖ Crearea unei vizualizări complexe. Dați un exemplu de operație LMD permisă pe vizualizarea respectivă și un exemplu de operație LMD nepermisă .....	25
❖ Formulați în limbaj natural și implementați în SQL: o cerere ce utilizează operația outer-join pe minimum 4 tabele, o cerere ce utilizează operația division și o cerere care implementează analiza top-n. ....	26
➤ Cerere ce utilizează operația outer-join pe minimum 4 tabele.....	26
➤ Cerere ce utilizează operația division: .....	27
➤ Cerere care implementează analiza top-n .....	28
❖ Două instrucțiuni select echivalente semantic, de comparat din punct de vedere a execuției (explicat plan de execuție).....	29
❖ Alegerea unor relații/join-uri din model și reprezentarea acestora într-o bază de date NoSql (MongoDb, Cassandra etc.).....	30
➤ Join în MongoDB între tabelul cu produse și recenzii.....	31
➤ Join în MongoDB între user_address, countries și regions.....	32
❖ Tranzacții: ilustrarea consistency levels in Oracle cu tranzacții care operează asupra modelului ales.....	34
➤ Dirty write.....	34
➤ Lost update.....	35

### Descrierea modelului real, a utilității acestuia și a regulilor de funcționare.

Sparkle este un magazin online de distribuire a jocurilor video, în format fizic și digital. Indivizii pot comanda jocuri cu sau fără un cont creat. Pentru a crea un cont, este necesar un nume de utilizator, parola, nume și prenume. În baza de date, acesta va primi un ID și se va cunoaște data la care utilizatorul a fost creat, iar în urma modificărilor asupra contului, va fi actualizată data la care au fost făcute modificările. Utilizatorul nu este obligat să introducă detalii privind adresa sa în momentul generării contului, însă dacă este plasată o comandă, acestuia îi va fi asociată o adresă și o metodă de plată.

Adresa va avea în baza de date un ID unic și îi va fi asociat ID-ul userului și ID-ul țării, pentru a ne asigura că nu vor fi 2 adrese cu aceleași date, pentru a putea prelua cu ușurință adresele fiecărui utilizator și pentru a putea ca un utilizator să își poată salva mai multe adrese asociate contului.

Metoda de plată va avea un ID unic și un ID al utilizatorului, tipul plății (PayPal, Card, Cash) și numărul contului (dacă este cazul).

Produsele vor avea un publisher, ca persoană de contact pentru compania noastră. Mai mult de atât, fiecare produs va avea unul sau mai multe genuri din care face parte, dar și o categorie: joc(GME), addOn (ADD) sau conținut în plus pentru joc (DLC).

Utilizatorii, după achiziție, vor putea lăsa review-uri, care vor conține o notă de la 1 la 5 alături de opinia personală și experiența cu jocul, dar și cu livrarea și interacțiunea cu site-ul.

### Prezentarea constrângerilor (restricții, reguli) impuse asupra modelului.

Un utilizator poate avea zero sau mai multe adrese asociate contului, la fel și în cazul metodei de plată.

Un utilizator poate avea zero sau mai multe recenzii lăsate. De asemenea, acesta poate avea zero sau mai multe comenzi. Produsele pot face parte din niciuna sau din mai multe comenzi, fiecare înregistrare din tabel având ID-ul unei comenzi și ID-ul unui produs.

Un produs are obligatoriu un publisher și numai unul, dar un publisher poate avea cel puțin un produs, sau mai multe.

Un produs poate avea unul sau mai multe genuri în care poate fi încadrat, dar poate avea o singură categorie din cele trei categorii disponibile (GME, ADD, DLC).

Produsele pot avea niciunul sau mai multe recenzii.

### Descrierea entităților, incluzând precizarea cheii primare.

ENTITATE	CHEIE PRIMARA	OBSERVATII
users	user_id	Persoana care are cont pe site, pentru a identifica fiecare individ.
user_address	user_address_id	Adresa unde a fost trimisă comanda sau factura.
user_payment	user_payment_id	Metoda de plată asociată fiecărui utilizator.
country	country_id	Țările asociate adreselor .
region	region_id	Regiuni asociate adreselor.
review	[user_id, product_id]	Recenzii oferite de către utilizatori.
orders	order_id	Comanda făcută de un individ.
product_from_order	[order_id, product_id]	Comanda poate avea mai multe produse, produs_din_comanda este folosit pentru asocierea fiecărei comenzi cu unul sau mai multe produse.
products	product_id	Produsele pe care compania noastră le vinde.
product_type	product_type_id	Categoria din care face parte fiecare produs.
publishers	publisher_id	Producătorul fiecărui produs, detalii de contact pentru companie.

genre	genre_id	Genul fiecărui joc.
product_genres	[product_id, genre_id]	Tabel asociativ pentru a asocia fiecare produs cu unul sau mai multe genuri.

Descrierea relațiilor, incluzând precizarea cardinalității acestora.

RELATIE	CARDINALITATE	OBSERVATII
are	users – user_address: one-to-many users – user_payment: one-to-many	Un utilizator poate avea mai multe adrese, dar o adresa va fi asociata cu un singur utilizator, la fel pentru payment.
oferă	users – reviews: one-to-many	Un utilizator poate oferi mai multe review-uri, iar fiecare review este asociat unui utilizator.
plasează	users – orders: one-to-many	Un utilizator poate plasa mai multe comenzi, iar fiecare comanda va fi asociată unui utilizator.
include	orders - products: many-to-many	O comanda poate fi asociată cu mai multe produse și un produs poate face parte din mai multe comenzi. Acestea vor fi unite prin tabelul asociativ product from order.

Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor.

ENTITATE: users

Atribut	Tip	Dimensiune / precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
user_id	number	6	Va fi creat cu un sequence și incrementat cu 1, pornind de la 1	
username	string	30		NOT NULL
password	string	50		NOT NULL
firstname	string	30		NOT NULL
lastname	string	30		NOT NULL
created	timestamp		current timestamp	
modified	timestamp			

ENTITATE: user\_address

Atribut	Tip	Dimensiune / precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
user_address_id	number	6	Va fi creat cu un sequence și incrementat cu 1, pornind de la 1	
user_id	number	6		NOT NULL
country_id	char	2		NOT NULL

state/province	string	50		NOT NULL
city	string	50		NOT NULL
street	string	100		NOT NULL
zip	string	20		NOT NULL
phone	string	30		NOT NULL

ENTITATE: user\_payment

Atribut	Tip	Dimensiune / precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
user_payment_id	integer	6		
user_id	integer	6		NOT NULL
payment_type	string	20		NOT NULL
account_no	string	20		NOT NULL

ENTITATE: countries

Atribut	Tip	Dimensiune / precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
country_id	char	2		
region_id	integer	1		NOT NULL
name	string	30		NOT NULL

ENTITATE: regions

Atribut	Tip	Dimensiune / precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
region_id	integer	1		
name	string	15		

ENTITATE: reviews

Atribut	Tip	Dimensiune / precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
user_id	integer	6		
product_id	integer	6		
opinion	string	2000		NOT NULL
rating	integer	1	1-5	NOT NULL

ENTITATE: product\_genres

Atribut	Tip	Dimensiune / precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
product_id	integer	6		
genre_id	integer	6		

ENTITATE: genres

Atribut	Tip	Dimensiune / precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
genre_id	integer	2		
name	string	20		NOT NULL

ENTITATE: product\_from\_order

Atribut	Tip	Dimensiune / precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
order id	integer	6		
product id	integer	6		
quantity	integer	3		NOT NULL

ENTITATE: order

Atribut	Tip	Dimensiune / precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
order id	integer	6		
user id	integer	6		
orderdate	timestamp	2	current timestamp	

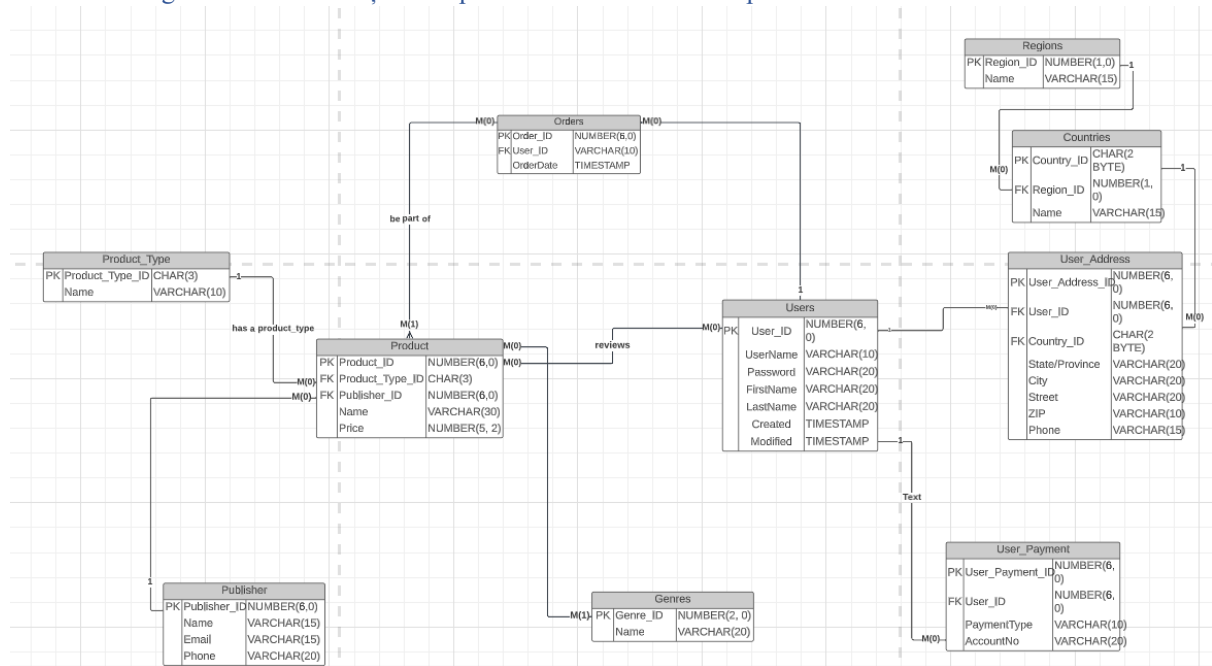
ENTITATE: product

Atribut	Tip	Dimensiune / precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
product id	integer	6		
product type id	char	3		
publisher id	integer	6		
name	string	30		NOT NULL
price	float	5/2		NOT NULL

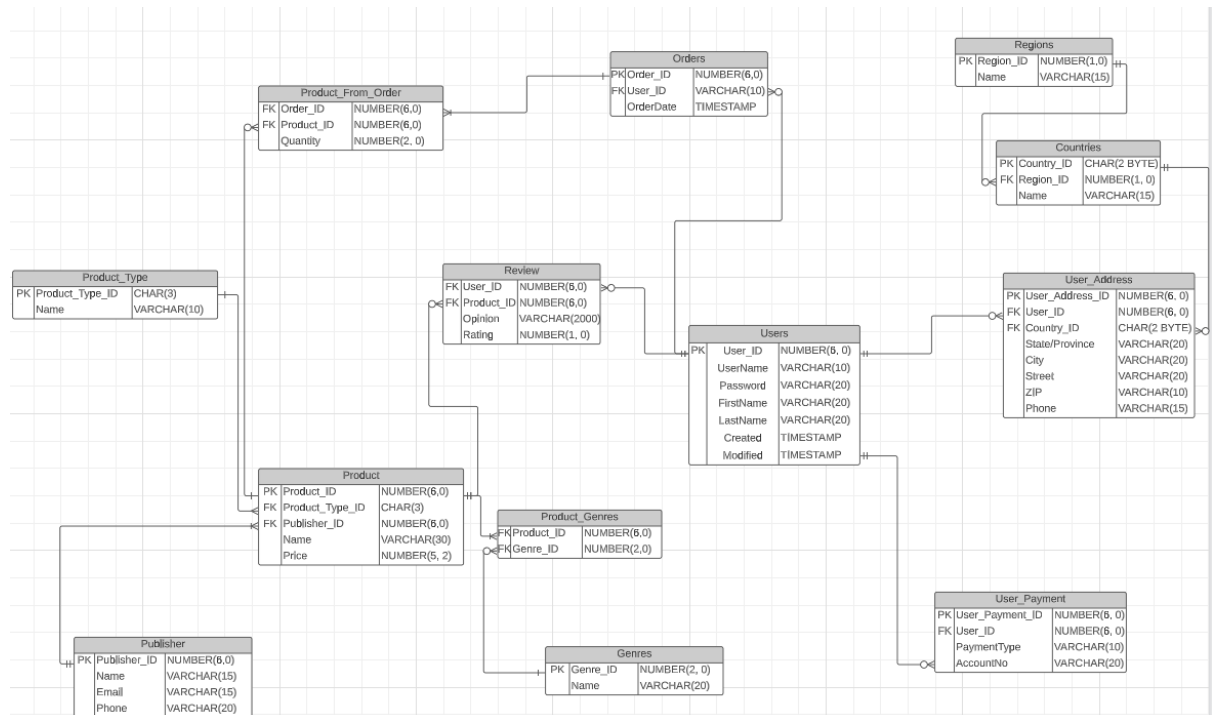
ENTITATE: publisher

Atribut	Tip	Dimensiune / precizie	Valori posibile si valori default	Observații, obligatoriu/opțional
publisher id	integer	6		
name	string	20		NOT NULL
email	string	50		NOT NULL
phone	string	20		NOT NULL

Realizarea diagramei entitate-relație corespunzătoare descrierii de la punctele 3-5.



Realizarea diagramei conceptuale corespunzătoare diagramei entitate-relație proiectate la punctul 6. Diagrama conceptuală trebuie să conțină minimum 6 tabele (fără considerarea subentităților), dintre care cel puțin un tabel asociativ.





Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectate la punctul 7.

USER(user\_id#, userName, password, firstName, lastName, createdAt, modifiedAt)  
USER\_PAYMENT(user\_payment\_id#, user\_id#, paymentType, accountNumber)  
USER\_ADDRESS(user\_address\_id#, user\_id#, country\_id#, state/county, city, street, zip, phone)  
COUNTRY(country\_id#, region\_id#, name)  
REGION(region\_id#, name)  
REVIEW([user\_id#, product\_id#], opinion, rating)  
PRODUCT(product\_id#, product\_type\_id#, publisher\_id#, name, price)  
PUBLISHER(publisher\_id#, name, email, phone)  
PRODUCT\_TYPE(product\_type\_id#, name)  
ORDER(order\_id#, user\_id#, orderDate)  
PRODUCT\_FROM\_ORDER(order\_id#, product\_id#, quantity)  
GENRES(genre\_id#, name)  
PRODUCT\_GENRE([product\_id#, genre\_id#])

Realizarea normalizării până la forma normală 3(FN1-FN3).

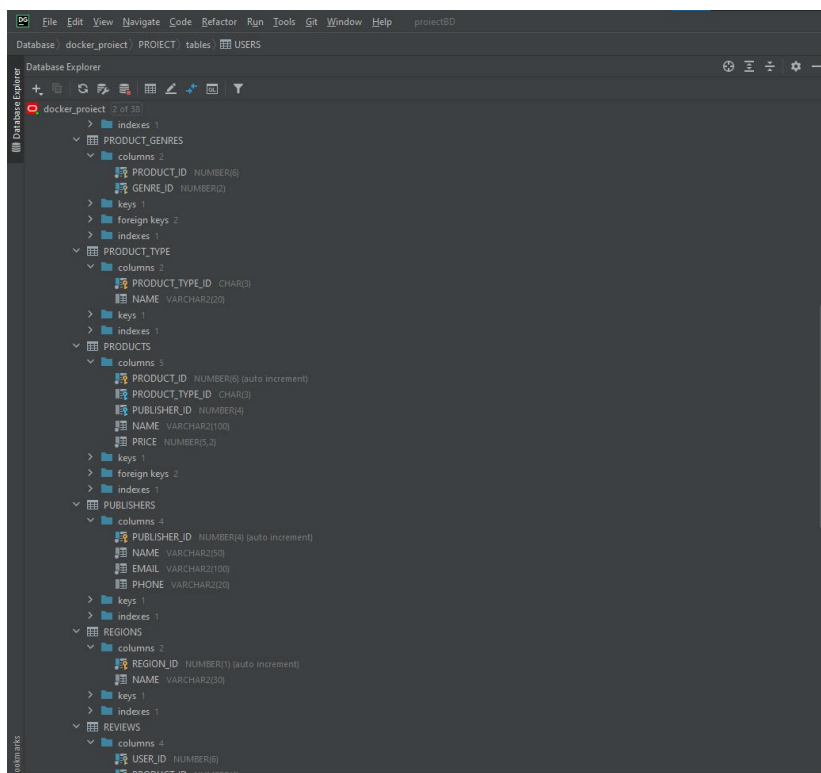
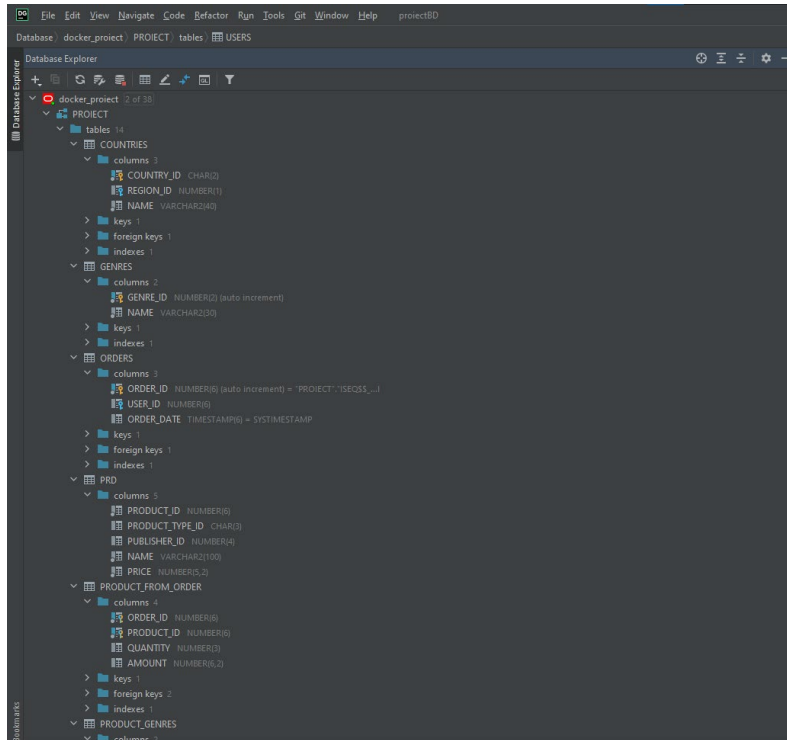
Pentru a exemplifica FN1, vom lua în considerare user\_address, users, user\_payment și reviews:

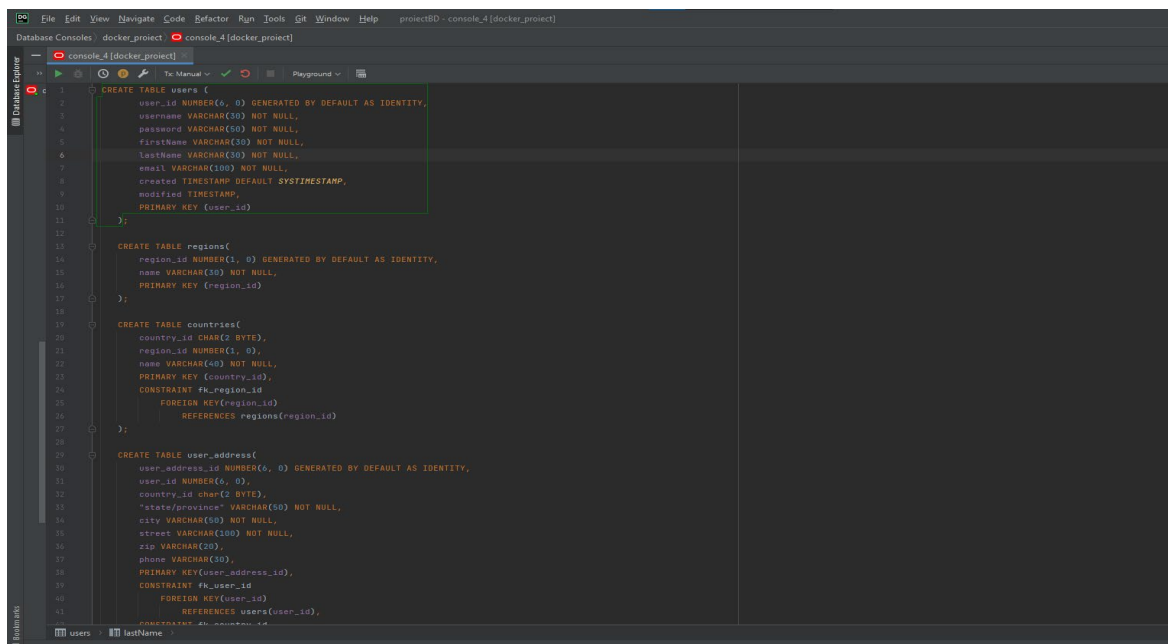
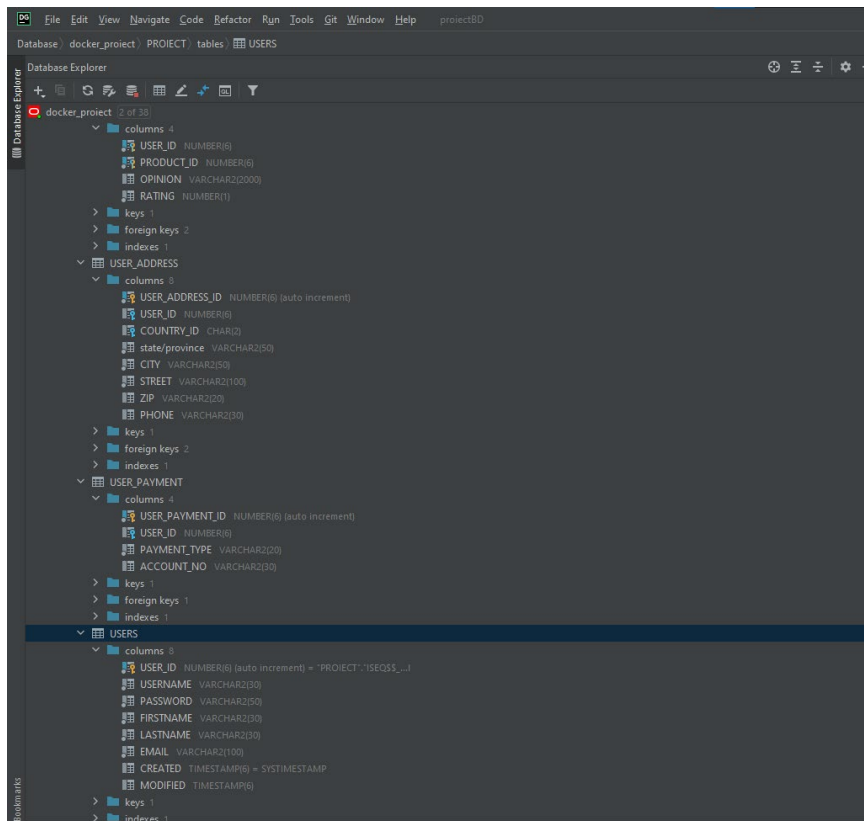
User(user\_id#, userName, password, firstName, lastName, createdAt, modifiedAt, paymentType, accountNumber, country\_id, state/county, city, street, zip, phone, product\_id, opinion, rating). Aici avem un exemplu unde vom avea parte de multa redundanță, de fiecare dată când un utilizator va adăuga o adresă sau va oferi o recenzie unui produs, trebuie adăugată o nouă coloană în tabel, unde se vor repeta datele utilizatorului, ocupând multă memorie.

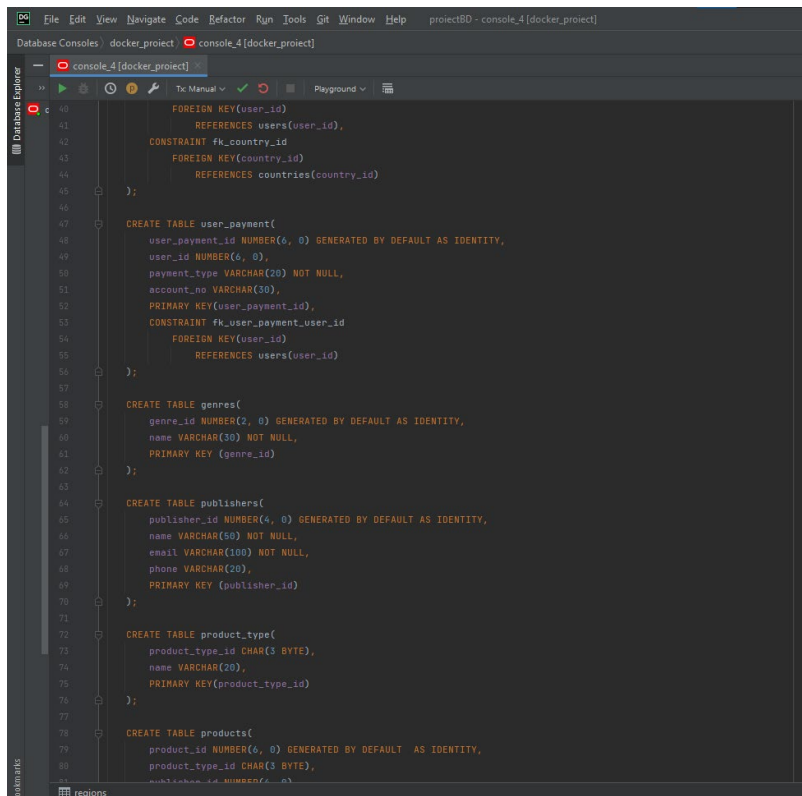
Pentru a aduce la FN3, cream noi tabele, unde vor avea fiecare cheia sa primară unică și eliminăm dependențele tranzitive.

USER(user\_id#, userName, password, firstName, lastName, createdAt, modifiedAt)  
USER\_PAYMENT(user\_payment\_id#, user\_id#, paymentType, accountNumber)  
USER\_ADDRESS(user\_address\_id#, user\_id#, country\_id#, state/county, city, street, zip, phone)  
REVIEWS(review\_id#, user\_id#, product\_id#, opinion, rating)

## Dovezi rulare cod intr-o baza de date ORACLE

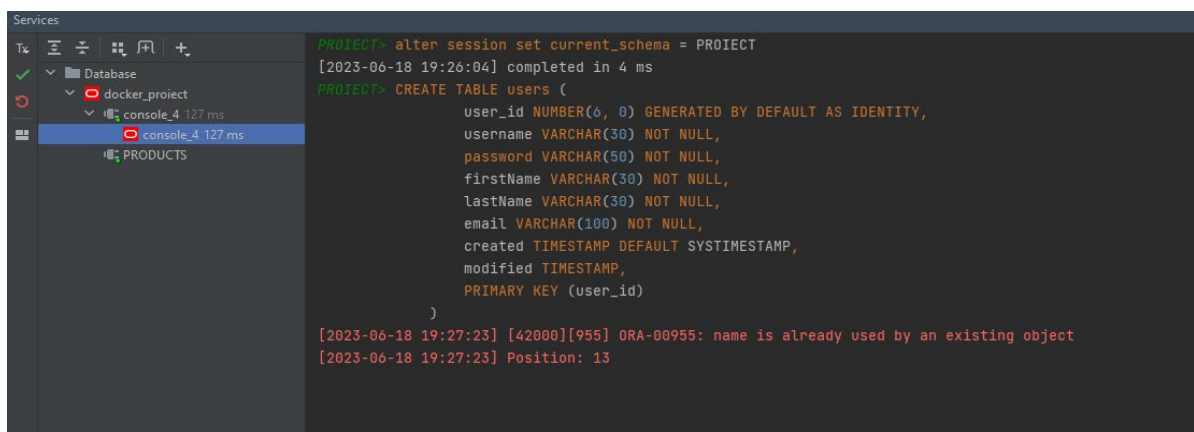






```
40 FOREIGN KEY(user_id)
41 REFERENCES users(user_id),
42 CONSTRAINT fk_country_id
43 FOREIGN KEY(country_id)
44 REFERENCES countries(country_id)
45 );
46
47 CREATE TABLE user_payment(
48 user_payment_id NUMBER(6, 0) GENERATED BY DEFAULT AS IDENTITY,
49 user_id NUMBER(6, 0),
50 payment_type VARCHAR(20) NOT NULL,
51 account_no VARCHAR(10),
52 PRIMARY KEY(user_payment_id),
53 CONSTRAINT fk_user_payment_user_id
54 FOREIGN KEY(user_id)
55 REFERENCES users(user_id)
56 );
57
58 CREATE TABLE genres(
59 genre_id NUMBER(2, 0) GENERATED BY DEFAULT AS IDENTITY,
60 name VARCHAR(10) NOT NULL,
61 PRIMARY KEY (genre_id)
62 );
63
64 CREATE TABLE publishers(
65 publisher_id NUMBER(4, 0) GENERATED BY DEFAULT AS IDENTITY,
66 name VARCHAR(50) NOT NULL,
67 email VARCHAR(100) NOT NULL,
68 phone VARCHAR(20),
69 PRIMARY KEY (publisher_id)
70 );
71
72 CREATE TABLE product_type(
73 product_type_id CHAR(3 BYTE),
74 name VARCHAR(20),
75 PRIMARY KEY(product_type_id)
76 );
77
78 CREATE TABLE products(
79 product_id NUMBER(6, 0) GENERATED BY DEFAULT AS IDENTITY,
80 product_type_id CHAR(3 BYTE),
81 publisher_id NUMBER(4, 0)
```

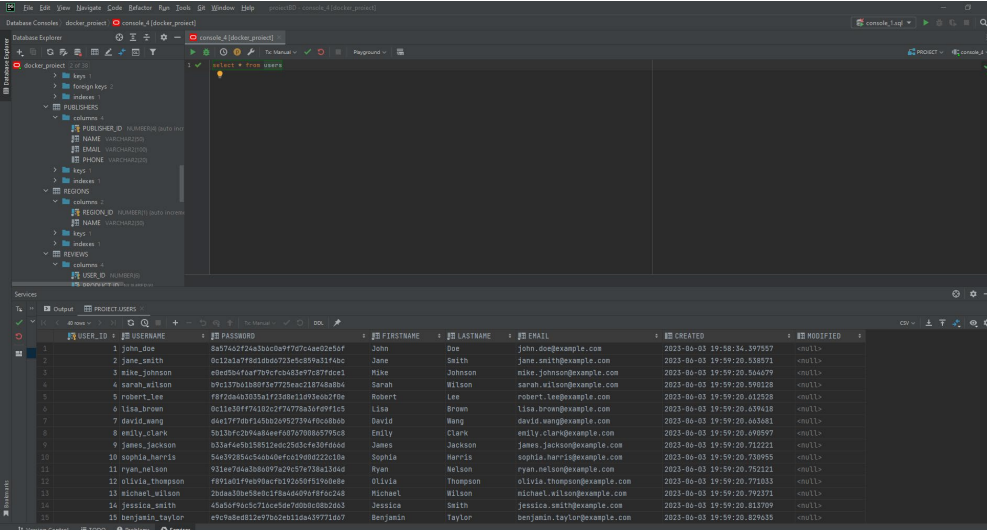
Menționez că baza de date a fost deja creată la momentul realizării printscreen-urilor, așa că rezultatele vor spune că există deja aceste tabele:



```
PROJECT> alter session set current_schema = PROIECT
[2023-06-18 19:26:04] completed in 4 ms
PROJECT> CREATE TABLE users (
    user_id NUMBER(6, 0) GENERATED BY DEFAULT AS IDENTITY,
    username VARCHAR(30) NOT NULL,
    password VARCHAR(50) NOT NULL,
    firstName VARCHAR(30) NOT NULL,
    lastName VARCHAR(30) NOT NULL,
    email VARCHAR(100) NOT NULL,
    created TIMESTAMP DEFAULT SYSTIMESTAMP,
    modified TIMESTAMP,
    PRIMARY KEY (user_id)
)
[2023-06-18 19:27:23] [42000][955] ORA-00955: name is already used by an existing object
[2023-06-18 19:27:23] Position: 13
```

Dovada faptului că datele sunt inserate în tabel:

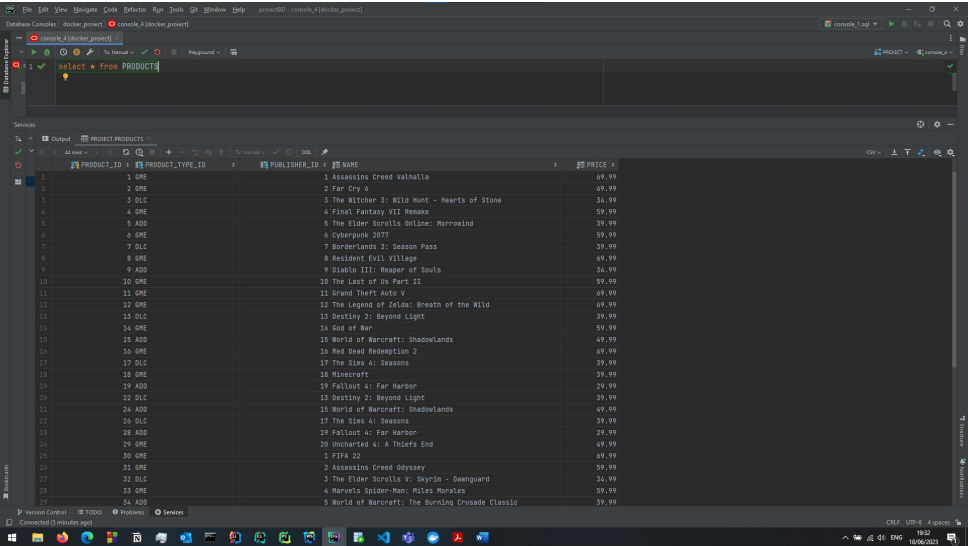
## Users



The screenshot shows a database console with the 'Users' table selected. The table contains 15 rows of user data, including user ID, username, password, first name, last name, email, creation date, and modification date.

USER_ID	USERNAME	PASSWORD	FIRSTNAME	LASTNAME	EMAIL	CREATED	MODIFIED
1	jane doe	6a1764742a3b3c0f979704ee02d6f	Jane	Doe	jane.doe@example.com	2023-06-03 19:59:20.597857	enull
2	jane smith	0c12a1776c1dd6d73d3e499a3174bc	Jane	Smith	jane.smith@example.com	2023-06-03 19:59:20.538371	enull
3	mike johnson	e8ed064a7b9cfca48e9c87c87f6ce1	Mike	Johnson	mike.johnson@example.com	2023-06-03 19:59:20.564679	enull
4	sarah wilson	89c137b01808f5e775fec218748db8a	Sarah	Wilson	sarah.wilson@example.com	2023-06-03 19:59:20.598128	enull
5	robert lee	f8976da3b30e1f330e1e1979a0a0f0e	Robert	Lee	robert.lee@example.com	2023-06-03 19:59:20.612520	enull
6	lisa brown	0c1e30ff774102c2f7477b0a16f09f1c5	Lisa	Brown	lisa.brown@example.com	2023-06-03 19:59:20.639418	enull
7	david wang	d4d17f7dbf1450b29527394f0cd8bb0	David	Wang	david.wang@example.com	2023-06-03 19:59:20.663681	enull
8	emily clark	5013bf7c20948d0ee6a87088a5795c3	Emily	Clark	emily.clark@example.com	2023-06-03 19:59:20.698597	enull
9	james jackson	633bf6c018155124e25c2f623f0f4d6	James	Jackson	james.jackson@example.com	2023-06-03 19:59:20.712221	enull
10	sophia harris	5a039284c4404d06f0c30a0e222c10a	Sophia	Harris	sophia.harris@example.com	2023-06-03 19:59:20.738955	enull
11	ryan nelson	931ee7d4a3b809f7a2c97e738a13d4d	Ryan	Nelson	ryan.nelson@example.com	2023-06-03 19:59:20.752121	enull
12	olivia thompson	f894e01f9eb90ac7319250f5190e08e	Olivia	Thompson	olivia.thompson@example.com	2023-06-03 19:59:20.771833	enull
13	michael wilson	20a3b30e1f330e1e1979a0a0f0e	Michael	Wilson	michael.wilson@example.com	2023-06-03 19:59:20.792371	enull
14	jessica smith	49a3b30e1f330e1e1979a0a0f0e	Jessica	Smith	jessica.smith@example.com	2023-06-03 19:59:20.813709	enull
15	benjamin taylor	e9e9a8ed12e7b2e0110a439772d67	Benjamin	Taylor	benjamin.taylor@example.com	2023-06-03 19:59:20.829635	enull

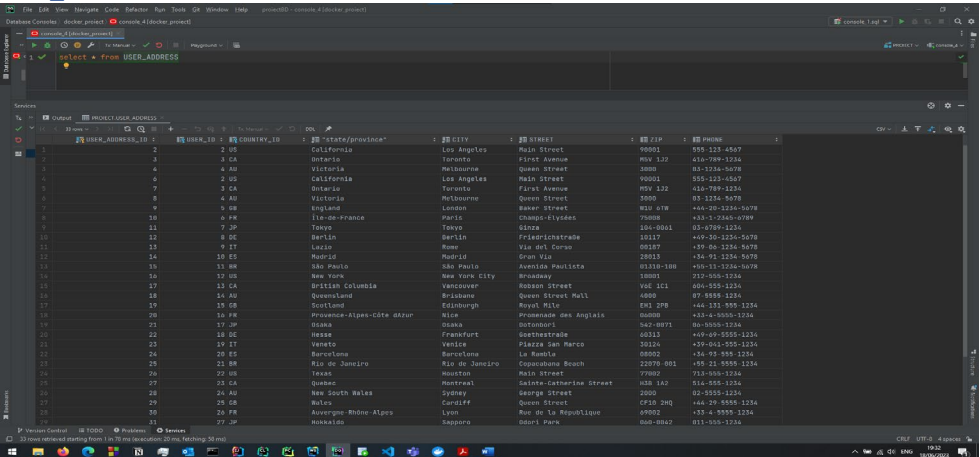
## Products



The screenshot shows a database console with the 'Products' table selected. The table contains 30 rows of product data, including product ID, product type ID, publisher ID, name, and price.

PRODUCT_ID	PRODUCT_TYPE_ID	PUBLISHER_ID	NAME	PRICE
1	ONE	1	Assassins Creed Valhalla	69.99
2	ONE	2	Far Cry 6	69.99
3	ONE	3	The Witcher 3: Wild Hunt - Hearts of Stone	36.99
4	ONE	4	Final Fantasy VII Remake	59.99
5	ADD	5	The Elder Scrolls Online: Morrowind	39.99
6	ONE	6	Cyberpunk 2077	59.99
7	ONE	7	Borderlands 3: Season Pass	39.99
8	ONE	8	Resident Evil Village	69.99
9	ADD	9	Diablo III: Reaper of Souls	36.99
10	ONE	10	The Last of Us Part II	59.99
11	ONE	11	Grand Theft Auto V	69.99
12	ONE	12	The Legend of Zelda: Breath of the Wild	69.99
13	ONE	13	Destiny 2: Beyond Light	39.99
14	ONE	14	God of War	59.99
15	ADD	15	World of Warcraft: Shadowlands	69.99
16	ONE	16	Red Dead Redemption 2	69.99
17	ONE	17	The Sims 4: Seasons	39.99
18	ADD	18	Fallout 4: Far Harbor	29.99
19	ONE	19	Destiny 2: Beyond Light	39.99
20	ONE	20	World of Warcraft: Shadowlands	69.99
21	ONE	21	The Sims 4: Seasons	39.99
22	ADD	22	Fallout 4: Far Harbor	29.99
23	ONE	23	Uncharted 4: A Thief's End	69.99
24	ONE	24	Far Cry 6	69.99
25	ONE	25	Assassins Creed Odyssey	59.99
26	ONE	26	The Elder Scrolls V: Skyrim - Dawnguard	36.99
27	ONE	27	Marvel's Spider-Man: Miles Morales	59.99
28	ONE	28	World of Warcraft: The Burning Crusade Classic	39.99
29	ONE	29	World of Warcraft: The Burning Crusade Classic	39.99
30	ONE	30	World of Warcraft: The Burning Crusade Classic	39.99

## User Address



The screenshot shows a database console with the 'User Address' table selected. The table contains 30 rows of user address data, including user ID, country ID, state/province, city, street, ZIP, and phone number.

USER_ADDRESS_ID	USER_ID	COUNTRY_ID	STATE/PROVINCE	CITY	STREET	ZIP	PHONE
1	2	2 US	California	Los Angeles	Main Street	90001	555-123-4567
2	3	1 CA	Ontario	Toronto	First Avenue	800-123	416-789-1234
3	4	4 AU	Victoria	Melbourne	Queen Street	3000	03-1234-5678
4	5	3 CA	California	Los Angeles	Main Street	90001	555-123-4567
5	6	3 CA	Ontario	Toronto	First Avenue	800-123	416-789-1234
6	8	4 AU	Victoria	Melbourne	Queen Street	3000	03-1234-5678
7	9	6 FR	England	London	Water Street	800-123	+44-20-1234-5678
8	10	6 FR	Île-de-France	Paris	Champs-Élysées	75008	+33-1-2345-6789
9	11	7 JP	Tokyo	Tokyo	Shinjuku	160-0001	03-8765-1234
10	12	13 SA	Berlin	Berlin	Friedrichstraße	10117	+49-30-1234-5678
11	13	9 IT	Lazio	Rome	Via del Corso	00187	+39-06-1234-5678
12	14	10 ES	Madrid	Madrid	Gran Vía	28013	+34-91-1234-5678
13	15	11 BR	São Paulo	São Paulo	Avenida Paulista	01318-100	+55-11-1234-5678
14	16	12 US	New York	New York City	Broadway	10001	212-555-1234
15	17	13 CA	British Columbia	Vancouver	Robson Street	V6B 1C1	604-555-1234
16	18	14 AU	Queensland	Brisbane	Queen Street Mall	4000	07-5555-1234
17	19	15 GB	Scotland	Edinburgh	Royal Mile	EH2 2JH	+44-131-555-1234
18	20	16 FR	Provence-Alpes-Côte d'Azur	Nice	Promenade des Anglais	06000	+33-4-9355-1234
19	21	17 JP	Osaka	Osaka	Dotonbori	542-0071	06-5555-1234
20	22	18 DE	Hesse	Frankfurt	Frankfurterstrasse	60612	+49-69-5555-1234
21	23	19 IT	Veneto	Venice	Piazza San Marco	30124	+39-041-555-1234
22	24	20 BR	Barcelonès	Barcelona	La Rambla	08002	+34-93-555-1234
23	25	21 BR	Rio de Janeiro	Rio de Janeiro	Copacabana Beach	22070-001	+55-21-5555-1234
24	26	22 US	Texas	Houston	Main Street	77002	713-555-1234
25	27	23 CA	Quebec	Montreal	Saint-Catherine Street	H3B 3A2	514-555-1234
26	28	24 AU	New South Wales	Sydney	George Street	2000	02-5555-1234
27	29	25 FR	Midi-Pyrénées	Toulouse	Quai de la Marquise	31000	+33-5-5555-1234
28	30	26 FR	Auvergne-Rhône-Alpes	Lyon	Rue de la République	69002	+33-4-5555-1234
29	31	27 JP	Kansai	Sapporo	Gorai Park	050-0002	011-555-1234

User\_payment

## Countries

## Regions

## Reviews

The screenshot shows a database console window with a query result for the 'Reviews' table. The query is `select * from reviews`. The result is a table with columns: `USER_ID`, `PRODUCT_ID`, `OPINION`, and `RATING`. The data is as follows:

USER_ID	PRODUCT_ID	OPINION	RATING
1	3	Amazing, totally recommend	5
2	15	Great product, exceeded my expectations!	4
3	7	Not bad, but could be better	3
4	32	Absolutely love it! Best purchase ever!	5
5	22	Average product, nothing special	2
6	10	Highly recommended, worth every penny	5
7	18	Disappointing quality, wouldn't buy again	1
8	28	Decent product, good value for money	4
9	6	Mediocre performance, expected more	3
10	37	Terrible product, avoid at all costs	1
11	14	Impressive product, highly satisfied!	5
12	29	Good value for the price, would recommend	4
13	5	Absolutely amazing! The best product I have ever bought!	5
14	10	This game is absolutely addictive! I can't stop playing!	5
15	22	Good game, but it lacks content after a while.	3
16	18	Incredible graphics and immersive gameplay. Highly recommended!	5
17	32	Disappointing game. It didn't live up to the hype.	2
18	9	This game is a masterpiece! The story and characters are amazing!	5
19	26	The gameplay feels repetitive and lacks innovation.	2
20	16	Excellent game! The mechanics are smooth, and the visuals are stunning.	5
21	35	Average game. It didn't really stand out.	3
22	12	I'm highly disappointed with this game. It's full of bugs and glitches.	1
23	21	This game offers great value for the price. It's highly enjoyable!	4
24	39	This game is exceptional! It's a must-play for any gamer!	5
25	3	I wouldn't recommend this game. It's unpolished and lacks depth.	1
26	31	I'm highly satisfied with this game. The gameplay is smooth and engaging!	5
27	40	Avoid this game. It's riddled with game-breaking bugs.	1
28	4	I'm satisfied with this game. The multiplayer is a lot of fun!	4
29	28	I had high expectations, but this game fell short. The story is lacking.	2

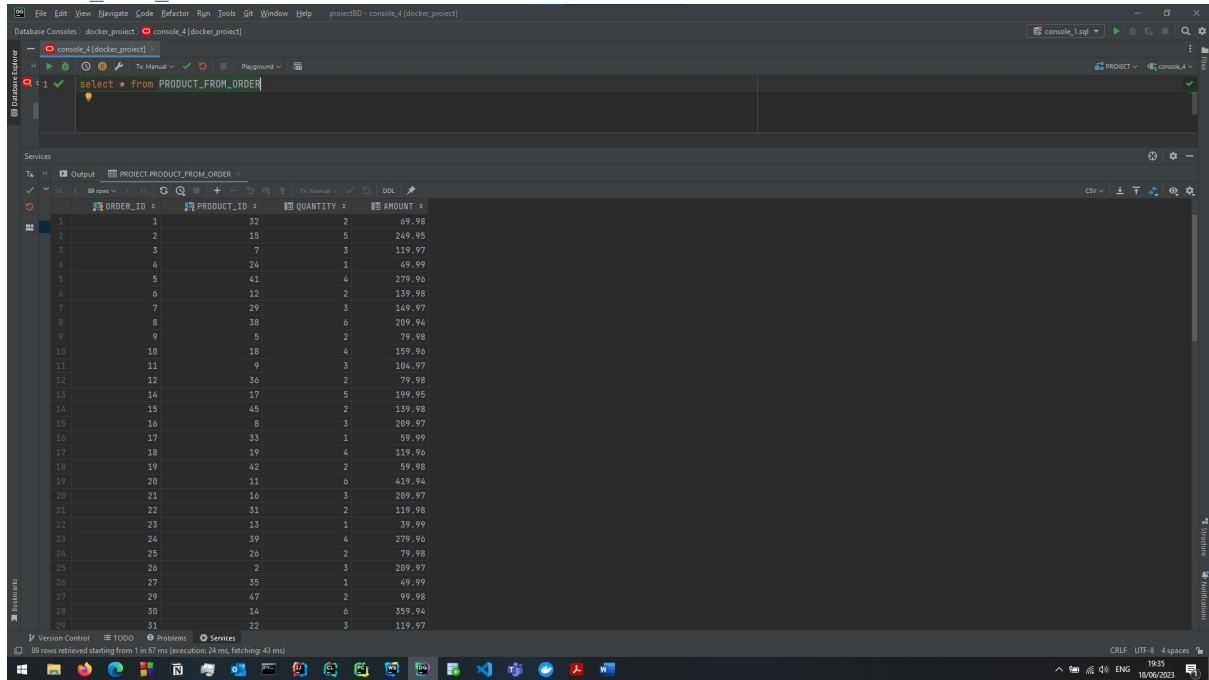
## Orders

The screenshot shows a database console window with a query result for the 'Orders' table. The query is `select * from orders`. The result is a table with columns: `ORDER_ID`, `USER_ID`, `ORDER_DATE`, and `ORDER_DATE_TIMESTAMP(6) (yyyy-MM-dd HH:mm:ss.fmmf)`. The data is as follows:

ORDER_ID	USER_ID	ORDER_DATE	ORDER_DATE_TIMESTAMP(6) (yyyy-MM-dd HH:mm:ss.fmmf)
1	23	2023-06-03 22:01:08	2023-06-03 22:01:08.506675
2	11	2023-06-03 22:01:08	2023-06-03 22:01:08.521439
3	31	2023-06-03 22:01:08	2023-06-03 22:01:08.549974
4	6	2023-06-03 22:01:08	2023-06-03 22:01:08.569569
5	39	2023-06-03 22:01:08	2023-06-03 22:01:08.590418
6	18	2023-06-03 22:01:08	2023-06-03 22:01:08.604850
7	8	2023-06-03 22:01:08	2023-06-03 22:01:08.621979
8	33	2023-06-03 22:01:08	2023-06-03 22:01:08.639184
9	21	2023-06-03 22:01:08	2023-06-03 22:01:08.651643
10	2	2023-06-03 22:01:08	2023-06-03 22:01:08.665094
11	15	2023-06-03 22:01:08	2023-06-03 22:01:08.681388
12	37	2023-06-03 22:01:08	2023-06-03 22:01:08.695494
13	24	2023-06-03 22:01:08	2023-06-03 22:01:08.708636
14	10	2023-06-03 22:01:08	2023-06-03 22:01:08.722186
15	28	2023-06-03 22:01:08	2023-06-03 22:01:08.736823
16	13	2023-06-03 22:01:08	2023-06-03 22:01:08.749015
17	34	2023-06-03 22:01:08	2023-06-03 22:01:08.762586
18	26	2023-06-03 22:01:08	2023-06-03 22:01:08.775915
19	1	2023-06-03 22:01:08	2023-06-03 22:01:08.789365
20	40	2023-06-03 22:01:08	2023-06-03 22:01:08.802373
21	5	2023-06-03 22:01:08	2023-06-03 22:01:08.814856
22	19	2023-06-03 22:01:08	2023-06-03 22:01:08.827702
23	30	2023-06-03 22:01:08	2023-06-03 22:01:08.840996
24	14	2023-06-03 22:01:08	2023-06-03 22:01:08.853442
25	35	2023-06-03 22:01:08	2023-06-03 22:01:08.866503
26	9	2023-06-03 22:01:08	2023-06-03 22:01:08.881754
27	27	2023-06-03 22:01:08	2023-06-03 22:01:08.895178
28	16	2023-06-03 22:01:08	2023-06-03 22:01:08.908893
29	3	2023-06-03 22:01:08	



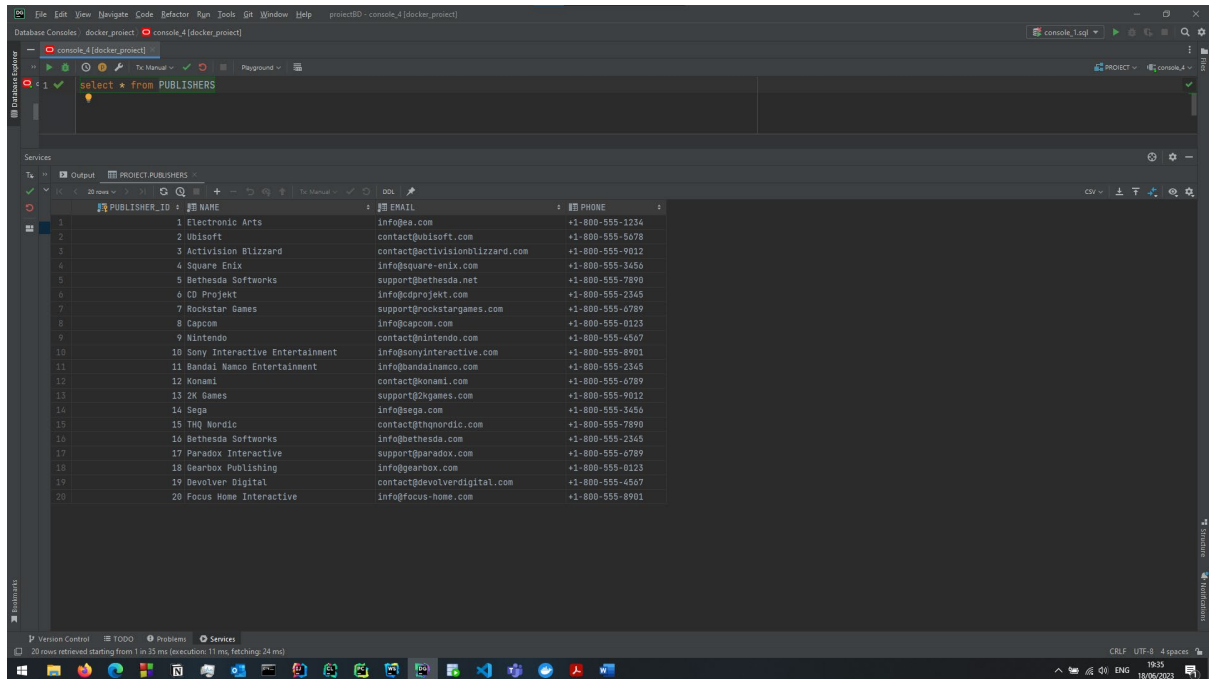
## Product\_from\_order



The screenshot shows a database IDE with a SQL query executed in the console. The query is `select * from PRODUCT_FROM_ORDER`. The results are displayed in a table with 29 rows. The table has four columns: `ORDER_ID`, `PRODUCT_ID`, `QUANTITY`, and `AMOUNT`. The data is as follows:

ORDER_ID	PRODUCT_ID	QUANTITY	AMOUNT
1	32	2	69.98
2	15	5	249.95
3	7	3	119.97
4	24	1	49.99
5	41	4	279.96
6	12	2	159.98
7	29	1	149.97
8	18	6	209.94
9	5	2	79.98
10	18	4	159.96
11	9	3	104.97
12	36	2	79.98
13	17	5	199.95
14	45	2	159.98
15	8	3	209.97
16	33	1	59.99
17	19	4	119.96
18	42	2	59.98
19	11	6	419.94
20	16	3	209.97
21	31	2	119.98
22	13	1	39.99
23	39	4	279.96
24	26	2	79.98
25	2	3	209.97
26	35	1	49.99
27	47	2	99.98
28	14	6	359.94
29	22	3	119.97

## Publishers



The screenshot shows a database IDE with a SQL query executed in the console. The query is `select * from PUBLISHERS`. The results are displayed in a table with 20 rows. The table has four columns: `PUBLISHER_ID`, `NAME`, `EMAIL`, and `PHONE`. The data is as follows:

PUBLISHER_ID	NAME	EMAIL	PHONE
1	Electronic Arts	info@ea.com	+1-800-555-1234
2	Ubisoft	contact@ubisoft.com	+1-800-555-5678
3	Activision Blizzard	contact@activisionblizzard.com	+1-800-555-9012
4	Square Enix	info@square-enix.com	+1-800-555-3456
5	Bethesda Softworks	support@bethesda.net	+1-800-555-7890
6	CD Projekt	info@cdprojekt.com	+1-800-555-2345
7	Rockstar Games	support@rockstargames.com	+1-800-555-6789
8	Capcom	info@capcom.com	+1-800-555-0123
9	Nintendo	contact@nintendo.com	+1-800-555-4567
10	Sony Interactive Entertainment	info@sonyinteractive.com	+1-800-555-8901
11	Bandai Namco Entertainment	info@bandainamco.com	+1-800-555-2345
12	Konami	contact@konami.com	+1-800-555-6789
13	2K Games	support@2kgames.com	+1-800-555-9012
14	Sega	info@sega.com	+1-800-555-3456
15	THQ Nordic	contact@thqnordic.com	+1-800-555-7890
16	Bethesda Softworks	info@bethesda.com	+1-800-555-2345
17	Paradox Interactive	support@paradox.com	+1-800-555-6789
18	Gearbox Publishing	info@gearbox.com	+1-800-555-0123
19	Devolver Digital	contact@devolverdigital.com	+1-800-555-4567
20	Focus Home Interactive	info@focus-home.com	+1-800-555-8901



## Genres

The screenshot shows an IDE window titled 'projectBD - console\_4 [docker\_project]'. The 'Database Explorer' on the left shows a connection to 'console\_4 [docker\_project]'. The 'Services' panel at the bottom shows the 'Output' window for 'PROJECT.GENRES'. The SQL query 'select \* from GENRES' is entered in the editor. The results are displayed in a table with 20 rows, showing 'GENRE\_ID' and 'NAME'.

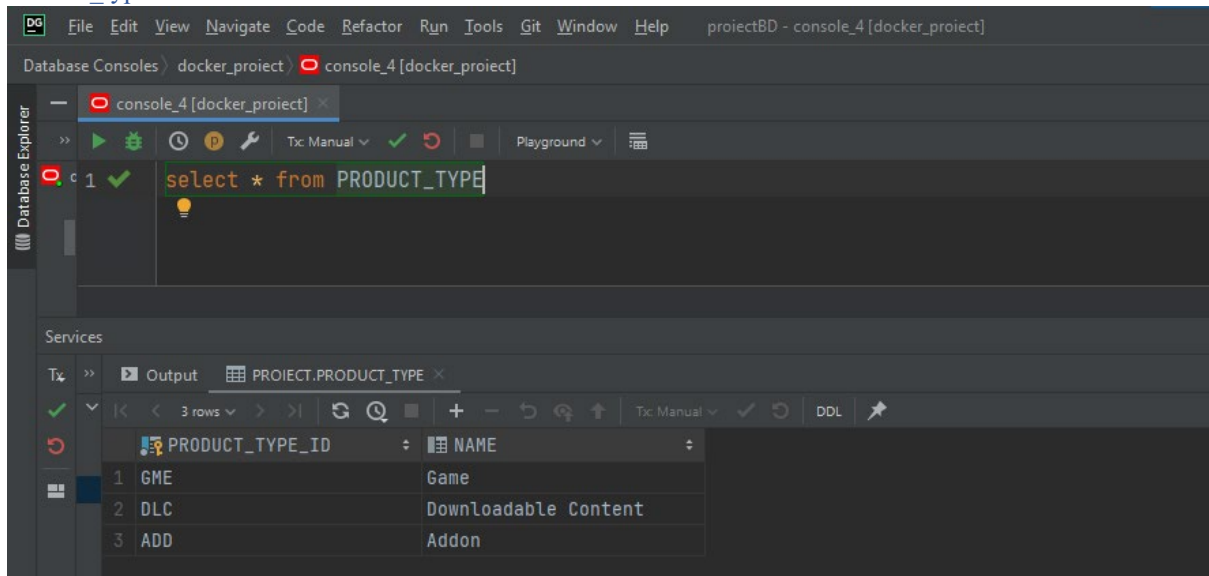
GENRE_ID	NAME
1	RPG
2	Horror
3	Open World
4	Stealth
5	MMORPG
6	Survival
7	Action-Adventure
8	Point-and-Click
9	Tower Defense
10	Battle Royale
11	Adventure
12	Role-Playing
13	Strategy
14	Simulation
15	Sports
16	Racing
17	Puzzle
18	Fighting
19	Shooter
20	Platformer

## Product\_genres

The screenshot shows an IDE window titled 'projectBD - console\_4 [docker\_project]'. The 'Database Explorer' on the left shows a connection to 'console\_4 [docker\_project]'. The 'Services' panel at the bottom shows the 'Output' window for 'PROJECT.PRODUCT\_GENRES'. The SQL query 'select \* from PRODUCT\_GENRES' is entered in the editor. The results are displayed in a table with 29 rows, showing 'PRODUCT\_ID' and 'GENRE\_ID'.

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

## Product\_type

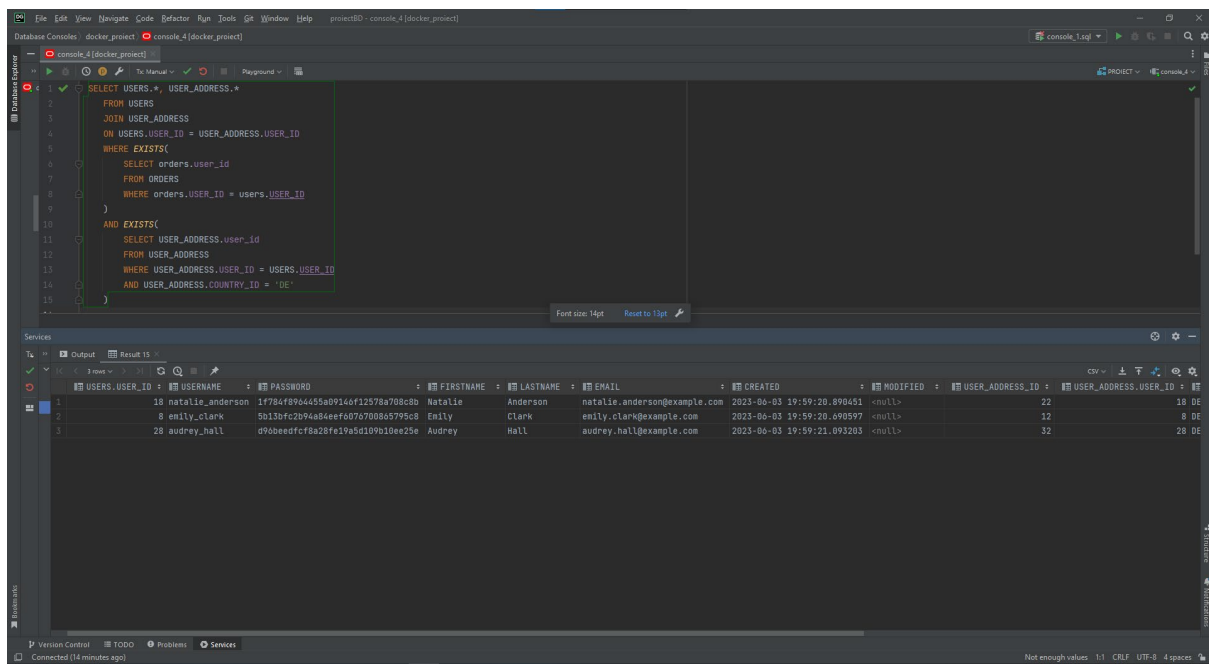


Formulați în limbaj natural și implementați 5 cereri SQL complexe ce vor utiliza, în ansamblul lor, următoarele elemente:

Subcereri sincronizate în care intervin cel puțin 3 tabele:

Afiseaza dintre toti userii, doar pe cei care au cel puțin o comanda, iar tara lor din adresa este germania

```
SELECT USERS.*, USER_ADDRESS.*
FROM USERS
JOIN USER_ADDRESS
ON USERS.USER_ID = USER_ADDRESS.USER_ID
WHERE EXISTS(
    SELECT orders.user_id
    FROM ORDERS
    WHERE orders.USER_ID = users.USER_ID
)
AND EXISTS(
    SELECT USER_ADDRESS.user_id
    FROM USER_ADDRESS
    WHERE USER_ADDRESS.USER_ID = USERS.USER_ID
    AND USER_ADDRESS.COUNTRY_ID = 'DE'
)
```



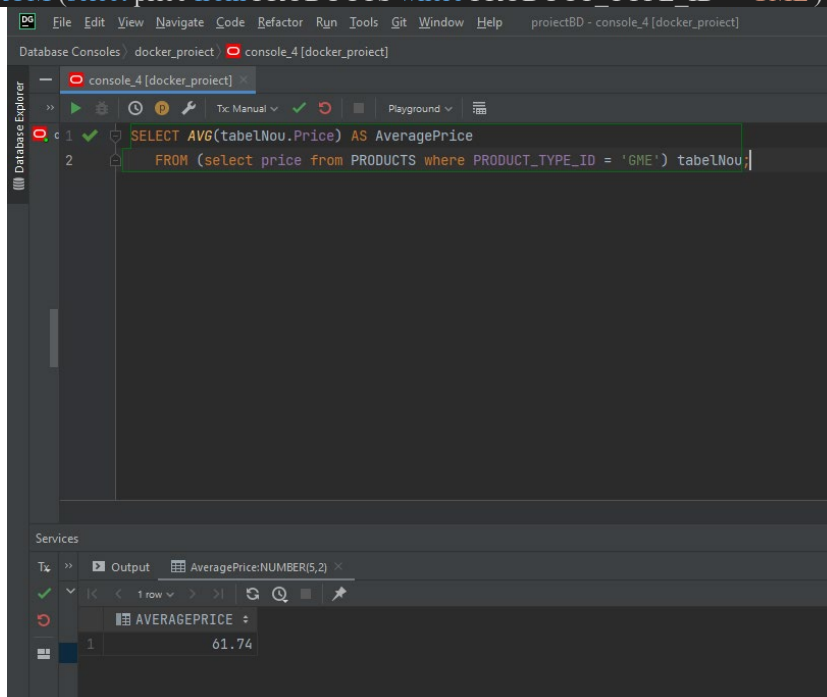
COUNTRY_ID	"state/province"	CITY	STREET	ZIP	PHONE
DE	Hesse	Frankfurt	GoethestraÙe	60313	+49-69-5555-1234
DE	Berlin	Berlin	FriedrichstraÙe	10117	+49-30-1234-5678
DE	North Rhine-Westphalia	Cologne	Hohe StraÙe	50667	+49-221-5555-1234

Subcereri nesincronizate în clauza FROM

Afișati pretul mediu al produselor de tip GME

SELECT AVG(tabelNou.Price) AS AveragePrice

FROM (select price from PRODUCTS where PRODUCT\_TYPE\_ID = 'GME') tabelNou;

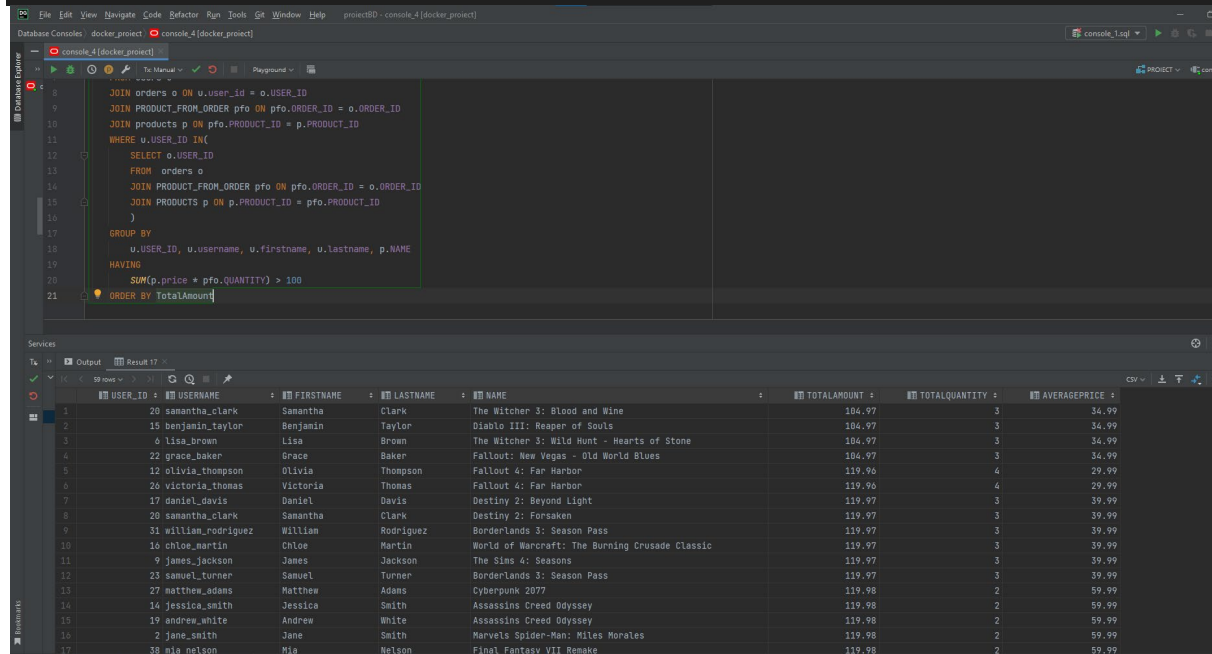


Grupări de date cu subcereri nesincronizate în care intervin cel puțin 3 tabele, funcții grup, filtrare la nivel de grupuri (în cadrul aceleiași cereri)

vrem să afișăm userii care au cumpărat ceva, ce anume au cumpărat cât a costat și câte au cumpărat, alături de prețul mediu. lista trebuie să fie ordonată după

-- total amount, și să fie afișati doar cei care au cheltuit mai mult de 100\$

```
SELECT u.USER_ID, u.username, u.firstname,
       u.lastname,
       p.NAME,
       SUM(p.price * pfo.QUANTITY) AS TotalAmount,
       SUM(pfo.QUANTITY) AS TotalQuantity,
       AVG(p.PRICE) AS AveragePrice
FROM users u
JOIN orders o ON u.user_id = o.USER_ID
JOIN PRODUCT_FROM_ORDER pfo ON pfo.ORDER_ID = o.ORDER_ID
JOIN products p ON pfo.PRODUCT_ID = p.PRODUCT_ID
WHERE u.USER_ID IN(
    SELECT o.USER_ID
    FROM orders o
    JOIN PRODUCT_FROM_ORDER pfo ON pfo.ORDER_ID = o.ORDER_ID
    JOIN PRODUCTS p ON p.PRODUCT_ID = pfo.PRODUCT_ID
)
GROUP BY
    u.USER_ID, u.username, u.firstname, u.lastname, p.NAME
HAVING
    SUM(p.price * pfo.QUANTITY) > 100
ORDER BY TotalAmount
```



The screenshot shows a database IDE with the following components:

- SQL Editor:** Contains the SQL query from the previous block.
- Services Panel:** Shows the execution results in a table format.

	USER_ID	USERNAME	FIRSTNAME	LASTNAME	NAME	TOTALAMOUNT	TOTALQUANTITY	AVERAGEPRICE
1	20	samantha_clark	Samantha	Clark	The Witcher 3: Blood and Wine	104.97	3	34.99
2	15	benjamin_taylor	Benjamin	Taylor	Diablo III: Reaper of Souls	104.97	3	34.99
3	8	lisa_brown	Lisa	Brown	The Witcher 3: Wild Hunt - Hearts of Stone	104.97	3	34.99
4	22	grace_baker	Grace	Baker	Fallout: New Vegas - Old World Blues	104.97	3	34.99
5	12	olivia_thompson	Olivia	Thompson	Fallout 4: Far Harbor	119.96	4	29.99
6	26	victoria_thomas	Victoria	Thomas	Fallout 4: Far Harbor	119.96	4	29.99
7	17	daniel_davis	Daniel	Davis	Destiny 2: Beyond Light	119.97	3	39.99
8	20	samantha_clark	Samantha	Clark	Destiny 2: Forsaken	119.97	3	39.99
9	31	william_rodriguez	William	Rodriguez	Borderlands 3: Season Pass	119.97	3	39.99
10	16	chloe_martin	Chloe	Martin	World of Warcraft: The Burning Crusade Classic	119.97	3	39.99
11	9	james_jackson	James	Jackson	The Sims 4: Seasons	119.97	3	39.99
12	23	samuel_turner	Samuel	Turner	Borderlands 3: Season Pass	119.97	3	39.99
13	27	matthew_adams	Matthew	Adams	Cyberpunk 2077	119.98	2	59.99
14	14	jessica_smith	Jessica	Smith	Assassins Creed Odyssey	119.98	2	59.99
15	19	andrew_white	Andrew	White	Assassins Creed Odyssey	119.98	2	59.99
16	2	jane_smith	Jane	Smith	Marvels Spider-Man: Miles Morales	119.98	2	59.99
17	28	mia_nelson	Mia	Nelson	Final Fantasy VII Remake	119.98	2	59.99

## Ordonări si utilizarea funcțiilor NVL și DECODE (in cadrul aceleiasi cereri)

-selectati userii care au comenzi, vezi ce au comandat

-- daca nu au lastModified, atunci pune timestamp din momentul query ului, iar tipurile de categorii le transformi in format lung, GME GAME, samd.

-- de asemenea vrem sa vedem datele comenzii, id ul, id ul produsului, cantitatea comandata, pretul unui produs si costul total pentru

SELECT

u.USER\_ID,

u.USERNAME,

NVL(u.MODIFIED, systimestamp) AS userModified,

p.NAME as productName,

DECODE(p.PRODUCT\_TYPE\_ID, 'GME', 'GAME', 'DLC', 'Downloadable Content', 'ADD', 'Addon') AS

productType,

o.ORDER\_ID,

p.PRODUCT\_ID,

pfo.QUANTITY,

p.PRICE,

SUM(p.PRICE \* pfo.QUANTITY) AS costTotal

FROM

users u

JOIN ORDERS o ON o.USER\_ID = u.USER\_ID

join PRODUCT FROM ORDER pfo ON pfo.ORDER\_ID = o.ORDER\_ID

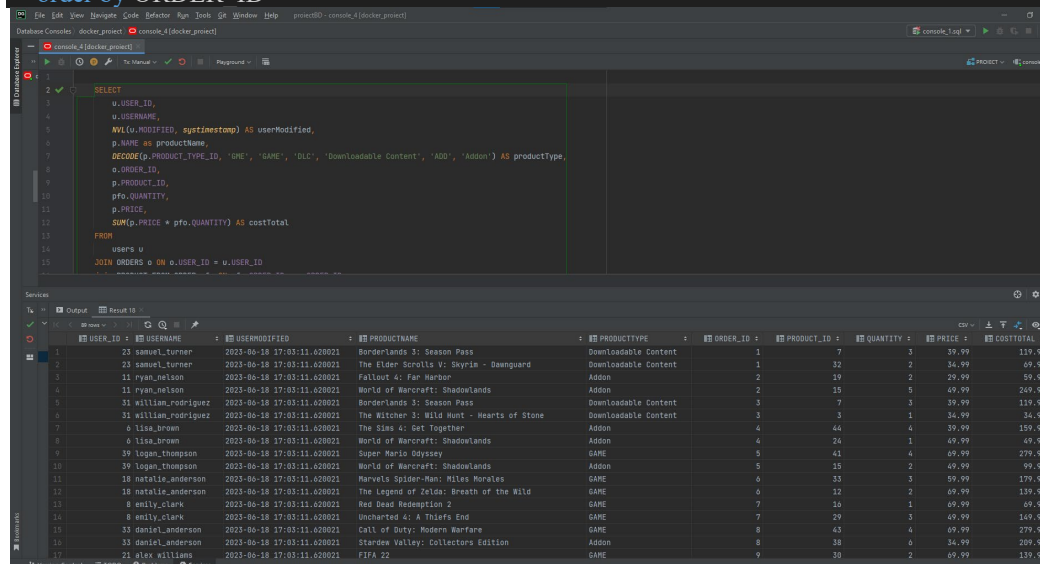
join PRODUCTS p ON p.PRODUCT\_ID = pfo.PRODUCT\_ID

GROUP BY u.USER\_ID, u.USERNAME, NVL(u.MODIFIED, systimestamp), p.NAME,

DECODE(p.PRODUCT\_TYPE\_ID, 'GME', 'GAME', 'DLC', 'Downloadable Content', 'ADD', 'Addon'),

o.ORDER\_ID, p.PRODUCT\_ID, pfo.QUANTITY, p.PRICE

order by ORDER\_ID



The screenshot shows an IDE with a SQL query editor and a results pane. The query is as follows:

```
SELECT
  u.USER_ID,
  u.USERNAME,
  NVL(u.MODIFIED, systimestamp) AS userModified,
  p.NAME as productName,
  DECODE(p.PRODUCT_TYPE_ID, 'GME', 'GAME', 'DLC', 'Downloadable Content', 'ADD', 'Addon') AS productType,
  o.ORDER_ID,
  p.PRODUCT_ID,
  pfo.QUANTITY,
  p.PRICE,
  SUM(p.PRICE * pfo.QUANTITY) AS costTotal
FROM
  users u
JOIN ORDERS o ON o.USER_ID = u.USER_ID
join PRODUCT FROM ORDER pfo ON pfo.ORDER_ID = o.ORDER_ID
join PRODUCTS p ON p.PRODUCT_ID = pfo.PRODUCT_ID
GROUP BY u.USER_ID, u.USERNAME, NVL(u.MODIFIED, systimestamp), p.NAME,
  DECODE(p.PRODUCT_TYPE_ID, 'GME', 'GAME', 'DLC', 'Downloadable Content', 'ADD', 'Addon'),
  o.ORDER_ID, p.PRODUCT_ID, pfo.QUANTITY, p.PRICE
order by ORDER_ID
```

The results pane displays a table with the following columns: USER\_ID, USERNAME, USERMODIFIED, PRODUCTNAME, PRODUCTTYPE, ORDER\_ID, PRODUCT\_ID, QUANTITY, PRICE, and COSTTOTAL. The data is sorted by ORDER\_ID.

USER_ID	USERNAME	USERMODIFIED	PRODUCTNAME	PRODUCTTYPE	ORDER_ID	PRODUCT_ID	QUANTITY	PRICE	COSTTOTAL
23	samuel_turner	2023-06-18 17:03:11.620021	Borderlands 3: Season Pass	Downloadable Content	1	7	3	39.99	119.97
23	samuel_turner	2023-06-18 17:03:11.620021	The Elder Scrolls V: Skyrim - Dawnguard	Downloadable Content	1	32	2	34.99	69.98
11	ryan_nelson	2023-06-18 17:03:11.620021	Fallout 4: Far Harbor	Addon	2	39	2	29.99	59.98
11	ryan_nelson	2023-06-18 17:03:11.620021	World of Warcraft: Shadowlands	Addon	2	35	5	49.99	249.95
31	william_rodriguez	2023-06-18 17:03:11.620021	Borderlands 3: Season Pass	Downloadable Content	3	7	3	39.99	119.97
31	william_rodriguez	2023-06-18 17:03:11.620021	The Witcher 3: Wild Hunt - Hearts of Stone	Downloadable Content	3	3	1	34.99	34.99
6	lisa_brown	2023-06-18 17:03:11.620021	The Sims 4: Get Together	Addon	4	44	4	39.99	159.96
6	lisa_brown	2023-06-18 17:03:11.620021	World of Warcraft: Shadowlands	Addon	4	34	1	49.99	49.99
39	logan_thompson	2023-06-18 17:03:11.620021	Super Mario Odyssey	GAME	5	41	4	49.99	279.96
39	logan_thompson	2023-06-18 17:03:11.620021	World of Warcraft: Shadowlands	Addon	5	35	2	49.99	99.98
18	natalie_anderson	2023-06-18 17:03:11.620021	Marvels Spider-Man: Miles Morales	GAME	6	33	3	59.99	179.97
18	natalie_anderson	2023-06-18 17:03:11.620021	The Legend of Zelda: Breath of the Wild	GAME	6	32	2	49.99	139.98
8	emily_clark	2023-06-18 17:03:11.620021	Red Dead Redemption 2	GAME	7	36	1	49.99	49.99
8	emily_clark	2023-06-18 17:03:11.620021	Uncharted 4: A Thief's End	GAME	7	29	3	49.99	149.97
33	daniel_anderson	2023-06-18 17:03:11.620021	Call of Duty: Modern Warfare	GAME	8	43	4	49.99	279.96
33	daniel_anderson	2023-06-18 17:03:11.620021	Stardev Valley: Collectors Edition	Addon	8	38	6	34.99	209.94
21	alex_williams	2023-06-18 17:03:11.620021	FIFA 22	GAME	9	30	2	49.99	139.98

Utilizarea a cel puțin 2 funcții pe șiruri de caractere, 2 funcții pe date calendaristice, a cel puțin unei expresii CASE

```
vrem sa unim prenumele si numele userilor intr o singura coloana si anume FULL_NAME, sa formatam tipul
timestamp in string si sa ne apara doar anul, luna si ziua
-- de asemenea vrem ziua saptamanii sa fie afisata, si avem 2 cazuri, daca luna cand a fost creat contul este a 6 a
sa apara iunie, daca e 7 sa apara iulie, altfel alta luna
-- iar coloana sa se numeasca created_month
SELECT
    UPPER(firstname) || ' ' || UPPER(lastname) AS full_name,
    TO_CHAR(CREATED, 'YYYY-MM-DD') AS formatted_created_date,
    TO_CHAR(CREATED, 'D') AS day_of_week,
    CASE
        WHEN EXTRACT(MONTH FROM CREATED) = 6 THEN 'JUNE'
        WHEN EXTRACT(MONTH FROM CREATED) = 7 THEN 'JULY'
        ELSE 'OTHER MONTH'
    END AS created_month
FROM USERS
```

Database Consoles > docker\_project > console\_4 [docker\_project]

```

1 SELECT
2     UPPER(firstname) || ' ' || UPPER(lastname) AS full_name,
3     TO_CHAR(CREATED, 'YYYY-MM-DD') AS formatted_created_date,
4     TO_CHAR(CREATED, 'D') AS day_of_week,
5     CASE
6         WHEN EXTRACT(MONTH FROM CREATED) = 6 THEN 'JUNE'
7         WHEN EXTRACT(MONTH FROM CREATED) = 7 THEN 'JULY'
8         ELSE 'OTHER MONTH'
9     END AS created_month
10 FROM USERS
11

```

day\_of\_week

Services

Output Result 19 ×

	FULL_NAME	FORMATTED_CREATED_DATE	DAY_OF_WEEK	CREATED_MONTH
1	JOHN DOE	2023-06-03	6	JUNE
2	JANE SMITH	2023-06-03	6	JUNE
3	MIKE JOHNSON	2023-06-03	6	JUNE
4	SARAH WILSON	2023-06-03	6	JUNE
5	ROBERT LEE	2023-06-03	6	JUNE
6	LISA BROWN	2023-06-03	6	JUNE
7	DAVID WANG	2023-06-03	6	JUNE
8	EMILY CLARK	2023-06-03	6	JUNE
9	JAMES JACKSON	2023-06-03	6	JUNE
10	SOPHIA HARRIS	2023-06-03	6	JUNE
11	RYAN NELSON	2023-06-03	6	JUNE
12	OLIVIA THOMPSON	2023-06-03	6	JUNE
13	MICHAEL WILSON	2023-06-03	6	JUNE
14	JESSICA SMITH	2023-06-03	6	JUNE
15	BENJAMIN TAYLOR	2023-06-03	6	JUNE
16	CHLOE MARTIN	2023-06-03	6	JUNE
17	DANIEL DAVIS	2023-06-03	6	JUNE

Utilizarea a cel puțin 1 bloc de cerere (clauza WITH)

- vrem sa facem un tabel addresses cu adresele existente din ITALIA, apoi din acel tabel temporar

-- sa selectam doar randurile cu orasul ROMA 'Rome', si sa ne afiseze numele intreg a persoanei care locuieste acolo, si adresa sa de email

```

WITH addresses AS(
    SELECT USER_ADDRESS_ID, COUNTRY_ID, "state/province", CITY, STREET, ZIP, PHONE,
    USER_ID
    FROM USER_ADDRESS
    WHERE COUNTRY_ID = 'IT'
),
addressFromItaly AS (
    select USER_ADDRESS_ID, COUNTRY_ID, "state/province", CITY, STREET, ZIP, PHONE, user_id
    from addresses
    where CITY = 'Rome'

```

```

)
select afi.user_address_id, aFI.country_id, aFI."state/province", aFI.city, aFI.street, aFI.zip, aFI.phone,
aFI.user_id,
    UPPER(u.firstname) || ' ' || UPPER(u.lastname) AS full_name,
    u.EMAIL
FROM addressFromItaly aFI
join users u ON u.USER_ID = afi.USER_ID

```

The screenshot shows an IDE with a SQL query editor and a results pane. The query is as follows:

```

WITH addresses AS(
    SELECT USER_ADDRESS_ID, COUNTRY_ID, "state/province", CITY, STREET, ZIP, PHONE, USER_ID
    FROM USER_ADDRESS
    WHERE COUNTRY_ID = 'IT'
),
    addressFromItaly AS (
        select USER_ADDRESS_ID, COUNTRY_ID, "state/province", CITY, STREET, ZIP, PHONE, user_id
        from addresses
        where CITY = 'Rome'
    )
select afi.user_address_id, aFI.country_id, aFI."state/province", aFI.city, aFI.street, aFI.zip, aFI.phone, aFI.user_id,
    UPPER(u.firstname) || ' ' || UPPER(u.lastname) AS full_name,
    u.EMAIL
FROM addressFromItaly aFI

```

The results pane shows a single row of data:

USER_ADDRESS_ID	COUNTRY_ID	state/province	CITY	STREET	ZIP	PHONE	USER_ID	FULL_NAME	EMAIL
13	IT	Lazio	Rome	Via del Corso	00187	+39-06-1234-5678	9	JAMES JACKSON	james.jackson@example.com

Implementarea a 3 operații de actualizare și de suprimare a datelor utilizând subcereri

```

-- ACTUALIZARE:
-- scadem pretul produselor de tip Addon cu 5 dolari

UPDATE PRODUCTS
SET PRICE = PRICE - 5
WHERE PRODUCT_TYPE_ID IN (
    SELECT PRODUCT_TYPE_ID
    FROM PRODUCT_TYPE
    WHERE PRODUCT_TYPE_ID = 'ADD'
)
AND PRICE > 10;

-- actualizam coloana amount din product_from_order, sa calculam cat a costat comanda la pretul pe care l avea
produsul in momentul cumpararii
UPDATE PRODUCT_FROM_ORDER pfo
SET AMOUNT = (
    SELECT p.price * pfo.QUANTITY
    FROM products p
    WHERE p.PRODUCT_ID = pfo.PRODUCT_ID
)

-- facem o reducere de 20% produselor care nu au facut parte din vreo comanda pana acum
UPDATE PRODUCTS
SET PRICE = PRICE * 0.8
WHERE PRODUCT_TYPE_ID IN (
    SELECT PRODUCT_TYPE_ID
    FROM PRODUCT_TYPE

```



```

    )
    AND PRODUCT_ID NOT IN (
        SELECT PRODUCT_ID
        FROM PRODUCT_FROM_ORDER
    )
-- SUPRIMARE
-- stergem datele de payment pentru userii a caror username incepe cu orice litera si se termina in mily, de ex:
Emily
DELETE FROM USER_PAYMENT
WHERE USER_ID IN (
    SELECT USER_ID
    FROM USERS
    WHERE USERNAME like '_mily'
)
-- stergem randurile care au account_no de lungime mai mare decat 15
DELETE FROM USER_PAYMENT
WHERE ACCOUNT_NO IN (
    SELECT ACCOUNT_NO
    FROM USER_PAYMENT
    WHERE LENGTH(account_no) > 15
)
-- stergem userii care au contul facut inainte de 2010, si care nu au nicio comanda plasata vreodata
DELETE FROM USERS
WHERE USER_ID IN (
    SELECT USER_ID
    FROM USERS
    WHERE TO_CHAR(CREATED, 'YYYY') < '2010'
)
AND USER_ID NOT IN (
    SELECT USER_ID
    FROM ORDERS
)

```

Crearea unei vizualizări complexe. Dați un exemplu de operație LMD permisă pe vizualizarea respectivă și un exemplu de operație LMD nepermisă

```

CREATE VIEW myComplexView AS
SELECT
    orders.order_id,
    users.user_id,
    users.USERNAME,
    products.name,
    PRODUCT_FROM_ORDER.quantity,
    PRODUCT_FROM_ORDER.amount
FROM ORDERS

```

```

JOIN USERS ON orders.USER_ID = USERS.USER_ID
JOIN PRODUCT_FROM_ORDER ON PRODUCT_FROM_ORDER.ORDER_ID = orders.ORDER_ID
JOIN PRODUCTS ON PRODUCTS.PRODUCT_ID = PRODUCT_FROM_ORDER.PRODUCT_ID
WHERE PRODUCT_FROM_ORDER.AMOUNT > 30
ORDER BY AMOUNT

```

-- exemplu de operatie LMD nepermisa:

-- fiind view complex, nu putem folosi insert pe mai multe dintre tabelele de baza deodata, trebuie doar unul care este protejat prin cheie

-- avand in vedere ca avem join, o instructiune poate afecta doar un singur tabel

```

insert into MYCOMPLEXVIEW (ORDER_ID, USER_ID, USERNAME, NAME, QUANTITY, AMOUNT)
values (1, 2, 'test', 'test', 3, 34)

```

-- exemplu de operatie LMD permisa:

-- afectam doar tabelul de baza product\_from\_order, din care provine coloana amount

```

update MYCOMPLEXVIEW

```

```

set amount = amount + 1

```

```

where user_id = 3

```

Formulați în limbaj natural și implementați în SQL: o cerere ce utilizează operația outer-join pe minimum 4 tabele, o cerere ce utilizează operația division și o cerere care implementează analiza top-n.

Cerere ce utilizează operația outer-join pe minimum 4 tabele

-- dorim sa afisam toti utilizatorii care au cel putin o recenzie, alaturi de produsele care au cel putin un review, sunt de tip GME, si sa nu ramana la final coloane null datorita outer joinului

-- datorita faptului ca avem mai multe tabele si este un outer join, vom avea multe coloane nule, asa ca la final ne asiguram ca user\_id nu e null, la fel si

-- product\_id, si afisati doar

```

select u.USER_ID, u.USERNAME, u.FIRSTNAME || ' ' || u.LASTNAME AS fullname,
       u.EMAIL, r.OPINION, r.RATING, p.product_id, p.PRODUCT_TYPE_ID, p.NAME,
       g.NAME AS genre,
       pbl.NAME AS publisher, pbl.EMAIL AS publisher_contact

```

```

from USERS u

```

```

RIGHT OUTER JOIN REVIEWS r ON u.USER_ID = r.USER_ID --da mi doar userii care au review uri

```

```

LEFT OUTER JOIN PRODUCTS p ON p.PRODUCT_ID = r.PRODUCT_ID --luam doar produsele care au review

```

```

FULL OUTER JOIN PRODUCT_GENRES pg ON pg.PRODUCT_ID = p.PRODUCT_ID --le luam pe toate pt ca toate produsele de tip GME au cel putin un gen asociat

```

```

FULL OUTER JOIN GENRES g ON g.GENRE_ID = pg.GENRE_ID --facem rost de numele genurilor

```

```

FULL OUTER JOIN PUBLISHERS pbl ON pbl.PUBLISHER_ID = p.PUBLISHER_ID

```

```

where u.USER_ID IS NOT NULL AND p.PRODUCT_ID IS NOT NULL AND p.PRODUCT_TYPE_ID = 'GME'

```

```

ORDER BY u.USERNAME

```

The screenshot shows a VS Code editor with a SQL query in the main editor and its results in the Services panel. The query is a complex JOIN query that retrieves user information, product details, and reviews. The results are displayed in a table with columns for user ID, username, full name, email, opinion, rating, product ID, product type, and game name.

```

-- product_id, si afisati doar
select u.USER_ID, u.USERNAME, u.FIRSTNAME || ' ' || u.LASTNAME AS fullname,
       u.EMAIL, r.OPINION, r.RATING, p.PRODUCT_ID, p.PRODUCT_TYPE_ID, p.NAME,
       g.NAME AS genre,
       pbl.NAME AS publisher, pbl.EMAIL AS publisher_contact
from USERS u
RIGHT OUTER JOIN REVIEWS r ON u.USER_ID = r.USER_ID --da mi doar userii care au review uri
LEFT OUTER JOIN PRODUCTS p ON p.PRODUCT_ID = r.PRODUCT_ID --luam doar produsele care au review
FULL OUTER JOIN PRODUCT_GENRES pg ON pg.PRODUCT_ID = p.PRODUCT_ID --le luam pe toate pt ca toate produsele de tip GME au cel putin un gen asociat
FULL OUTER JOIN GENRES g ON g.GENRE_ID = pg.GENRE_ID --facem rost de numele genurilor
FULL OUTER JOIN PUBLISHERS pbl ON pbl.PUBLISHER_ID = p.PUBLISHER_ID
where u.USER_ID IS NOT NULL AND p.PRODUCT_ID IS NOT NULL AND p.PRODUCT_TYPE_ID = 'GME'
ORDER BY u.USERNAME

```

USER_ID	USERNAME	FULLNAME	EMAIL	OPINION	RATING	PRODUCT_ID	PRODUCT_TYPE_ID	NAME
32	abigail_green	Abigail Green	abigail.green@example.com	Disappointing game. It didnt live up to the hype.	2	14	GME	God of War
32	abigail_green	Abigail Green	abigail.green@example.com	Disappointing game. It didnt live up to the hype.	2	14	GME	God of War
32	abigail_green	Abigail Green	abigail.green@example.com	Disappointing game. It didnt live up to the hype.	2	14	GME	God of War
15	benjamin_taylor	Benjamin Taylor	benjamin.taylor@example.com	Great product, exceeded my expectations!	4	8	GME	Resident Evil Village
15	benjamin_taylor	Benjamin Taylor	benjamin.taylor@example.com	Great product, exceeded my expectations!	4	8	GME	Resident Evil Village
15	benjamin_taylor	Benjamin Taylor	benjamin.taylor@example.com	Great product, exceeded my expectations!	4	8	GME	Resident Evil Village
16	chloe_martin	Chloe Martin	chloe.martin@example.com	Excellent game! The mechanics are smooth, and the visuals are stunning.	5	6	GME	Cyberpunk 2077
16	chloe_martin	Chloe Martin	chloe.martin@example.com	Excellent game! The mechanics are smooth, and the visuals are stunning.	5	6	GME	Cyberpunk 2077
16	chloe_martin	Chloe Martin	chloe.martin@example.com	Excellent game! The mechanics are smooth, and the visuals are stunning.	5	6	GME	Cyberpunk 2077
16	chloe_martin	Chloe Martin	chloe.martin@example.com	Excellent game! The mechanics are smooth, and the visuals are stunning.	5	6	GME	Cyberpunk 2077
33	daniel_anderson	Daniel Anderson	daniel.anderson@example.com	Dont waste your money on this game. Its full of microtransactions.	1	39	GME	God of War Ragnarok
33	daniel_anderson	Daniel Anderson	daniel.anderson@example.com	Dont waste your money on this game. Its full of microtransactions.	1	39	GME	God of War Ragnarok
33	daniel_anderson	Daniel Anderson	daniel.anderson@example.com	Dont waste your money on this game. Its full of microtransactions.	1	39	GME	God of War Ragnarok
8	emily_clark	Emily Clark	emily.clark@example.com	An amazing experience! The gameplay mechanics are top-notch.	5	31	GME	Assassins Creed Odyssey
8	emily_clark	Emily Clark	emily.clark@example.com	An amazing experience! The gameplay mechanics are top-notch.	5	31	GME	Assassins Creed Odyssey
24	emma_rodriguez	Emma Rodriguez	emma.rodriguez@example.com	Poor quality game. The controls are clunky and unresponsive.	1	37	GME	Resident Evil 2 Remake

NAME	GENRE	PUBLISHER	PUBLISHER_CONTACT
God of War	Adventure	Sega	info@sega.com
God of War	Fighting	Sega	info@sega.com
God of War	RPG	Sega	info@sega.com
Resident Evil Village	Stealth	Capcom	info@capcom.com
Resident Evil Village	Horror	Capcom	info@capcom.com
Resident Evil Village	Adventure	Capcom	info@capcom.com
Cyberpunk 2077	Adventure	CD Projekt	info@cdprojekt.com
Cyberpunk 2077	Role-Playing	CD Projekt	info@cdprojekt.com
Cyberpunk 2077	Open World	CD Projekt	info@cdprojekt.com
Cyberpunk 2077	Shooter	CD Projekt	info@cdprojekt.com
God of War Ragnarok	Adventure	Sony Interactive Entertainment	info@sonyinteractive.com
God of War Ragnarok	Action-Adventure	Sony Interactive Entertainment	info@sonyinteractive.com
God of War Ragnarok	RPG	Sony Interactive Entertainment	info@sonyinteractive.com
Assassins Creed Odyssey	Adventure	Ubisoft	contact@ubisoft.com
Assassins Creed Odyssey	RPG	Ubisoft	contact@ubisoft.com
Resident Evil 2 Remake	MMORPG	Capcom	info@capcom.com

Cerere ce utilizează operația division:

- afisati id ul tuturor comenzilor care contin toate produsele cu pretul mai mare decat 70 dolari. cum nu exista produse cu pretul mai mare de 70 dolari,  
 -- vor fi afisate toate comenzile  
 -- am folosit metoda 1 pentru a utiliza operatia division, in care ne folosim de 2 not exists

```

SELECT DISTINCT ORDER_ID
FROM PRODUCT_FROM_ORDER a
WHERE NOT EXISTS(
  SELECT 1
  FROM products p
  WHERE price > 70
  AND NOT EXISTS(
    SELECT 'x'

```

```

FROM PRODUCT_FROM_ORDER b
WHERE p.PRODUCT_ID = b.PRODUCT_ID
AND b.ORDER_ID = a.ORDER_ID
)
)

```

The screenshot shows a database IDE with a SQL query editor and a results pane. The query editor contains a SQL query that uses a nested `NOT EXISTS` clause to filter orders. The results pane shows a table with 58 rows, where the first column is labeled `ORDER\_ID` and the values range from 1 to 18.

```

-- am folosit metoda 1 pentru a utiliza operatia division, in care ne folosim de 2 not exists
SELECT DISTINCT ORDER_ID
FROM PRODUCT_FROM_ORDER a
WHERE NOT EXISTS(
    SELECT 1
    FROM products p
    WHERE price > 70
    AND NOT EXISTS(
        SELECT 'x'
        FROM PRODUCT_FROM_ORDER b
        WHERE p.PRODUCT_ID = b.PRODUCT_ID
        AND b.ORDER_ID = a.ORDER_ID
    )
)

```

ORDER_ID
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

Cerere care implementează analiza top-n

```

-- dorim sa afisam primele 10 randuri din tabelul cu produsele sortate dupa pret
-- pentru asta avem la dispozitie 2 metode, fie sortam prima data tot tabelul cu un subquery
-- iar pe urma selectam din el doar primele 10 randuri:
SELECT * FROM (
    SELECT * FROM PRODUCTS
    ORDER BY PRICE
) WHERE ROWNUM <= 10

```

```
-- fie folosim fetch
select * from PRODUCTS
order by price
fetch first 10 rows only
```

Două instrucțiuni select echivalente semantic, de comparat din punct de vedere a execuției (explicat plan de execuție)

```
SELECT * FROM (
  SELECT * FROM PRODUCTS
  ORDER BY PRICE
) WHERE ROWNUM <= 10
```

```
-- Aceasta interogare ne rezulta acest plan de executie:
-- Plan hash value: 101034194
```

```
-----
-- | Id | Operation          | Name      | Rows | Bytes | Cost (%CPU)| Time     |
-----
-- | 0 | SELECT STATEMENT    |           |      |      |      |          | 00:00:01 |
-- |* 1 | COUNT STOPKEY       |           |      |      |      |          |          |
-- | 2 | VIEW                |           | 44   | 4224 | 4 (25)| 00:00:01 |
-- |* 3 | SORT ORDER BY STOPKEY|           |      |      | 4 (25)| 00:00:01 |
-- | 4 | TABLE ACCESS FULL  | PRODUCTS  | 44   | 1760 | 3 (0)| 00:00:01 |
-----
```

```
-- Predicate Information (identified by operation id):
-----
```

```
-- 1 - filter(ROWNUM<=10)
-- 3 - filter(ROWNUM<=10)
```

```
-- TOTAL BYTES UTILIZATI PENTRU INTEROGARE: 8704
```

```
-- //////////
```

```
select * from PRODUCTS
order by price
fetch first 10 rows only
```

```
-- Interogarea de mai sus, are acest plan de executie:
-- Plan hash value: 2063928979
```

```
-----
-- | Id | Operation          | Name      | Rows | Bytes | Cost (%CPU)| Time     |
-----
-- | 0 | SELECT STATEMENT    |           |      |      |      |          | 00:00:01 |
```

```

-- |* 1 | VIEW          |      | 10 | 1220 | 4 (25)| 00:00:01 |
-- |* 2 | WINDOW SORT PUSHED RANK|      | 44 | 1760 | 4 (25)| 00:00:01 |
-- | 3 | TABLE ACCESS FULL   | PRODUCTS | 44 | 1760 | 3 (0)| 00:00:01 |
-----

-- Predicate Information (identified by operation id):
-----

-- " 1 - filter("from$_subquery$_002"."rowlimit_$$_rownumber"<=10)"
-- " 2 - filter(ROW_NUMBER() OVER ( ORDER BY ""PRODUCTS"".""PRICE""<=10)"

-- TOTAL BYTES UTILIZATI PENTRU INTEROGARE: 5960

-- Interogarea cu subcerere incepe cu un acces la tabelul Products (4), apoi sorteaza, (*) ne arata ca la acel pas se creeaza o filtrare
-- pe urma creeaza o vizualizare, iar in punctul 1 numara din nou si filteaza, conditia WHERE ROWNUM <= 10. In final, sunt afisate randurile ramase





-- Interogarea cu fetch incepe cu un acces la tabelul Products(3), apoi direct se creeaza filtrarea prin order by price. in pasul (1),
-- putem observa deja ca mai avem doar 10 randuri, deci a fost facut si fetchul

-- Ca diferenta intre ele, observam ca este prezis faptul ca interogarea cu fetch ar folosi mai putina memorie, deoarece avem un pas in minus.

```

Alegerea unor relații/join-uri din model și reprezentarea acestora într-o bază de date NoSql (MongoDb, Cassandra etc.)

Pentru început, am instalat în docker imaginea pentru MongoDB.

	Name	Image	Status	Port(s)	Last started	Actions
	<a href="#">mongoDB</a> c65a67e85dfc	<a href="#">mongo:latest</a>	Exited	27017:27017	8 days ago	  

Pe urmă, cu ajutorul unor fișiere CSV exportate din DataGrip, am creat colecțiile cu ajutorul Mongo Compass. Cu ajutorul Mongoose am creat joinurile între următoarele tabele:

Join în MongoDB intre tabelul cu produse și recenzii.

MongoDB Compass - localhost:27017/proiectDB.products

Connect Edit View Collection Help

localhost:27017

Documents  
proiectDB.produ...

My Queries

Databases

Search

admin

config

local

proiectDB

- countries
- genres
- orders
- product\_from\_order
- product\_genres
- product\_type
- products**
- publishers
- regions
- reviews
- user\_address
- user\_payment
- users

proiectDB.products

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find More Options

ADD DATA EXPORT DATA

1 - 20 of 44

```
{
  "_id": ObjectId('647d7be2da1853984cb3dd57'),
  "PRODUCT_ID": 1,
  "PRODUCT_TYPE_ID": "GHE",
  "PUBLISHER_ID": 1,
  "NAME": "Assassins Creed Valhalla",
  "PRICE": 69.99,
  "REVIEWS": Array
    0: Object
      "_id": ObjectId('647d7bb6da1853984cb3dd1c'),
      "USER_ID": 37,
      "OPINION": "Terrible product, avoid at all costs",
      "RATING": 1
    1: Object
      "_id": ObjectId('647d7bb6da1853984cb3dd2b'),
      "USER_ID": 3,
      "OPINION": "I wouldnt recommend this game. Its unpolished and lacks depth.",
      "RATING": 1
  }
}
```

```
{
  "_id": ObjectId('647d7be2da1853984cb3dd58'),
  "PRODUCT_ID": 2,
  "PRODUCT_TYPE_ID": "GHE",
  "PUBLISHER_ID": 2,
  "NAME": "Far Cry 6",
  "PRICE": 69.99
}
```

```
const { MongoClient } = require('mongodb');

async function main() {
  const url = "mongodb://localhost:27017";
  const client = new MongoClient(url, { useUnifiedTopology: true });

  try {
    await client.connect();
    console.log('Connected successfully to server');

    const db = client.db('proiectDB');
    const productsCollection = db.collection('products');

    await productsCollection.updateMany({}, {
      $set: {REVIEWS: [] }
    }); //cream un nou array in care sa introducem review urile

    let products = await productsCollection.find().toArray();

    const reviewsCollection = db.collection('reviews');
    const reviews = await reviewsCollection.find().toArray();

    for(let review of reviews){ //ne uitam la fiecare review
      let pID = review.PRODUCT_ID; //ii salvam id ul produsului
      review.PRODUCT_ID = null; // ii golim valoarea
      delete review.PRODUCT_ID; //stergem obiectul din review pentru a l putea introduce in products fara
      product ID
    }
  } catch (err) {
    console.error('Error connecting to MongoDB: ', err);
  }
}
```

```

    await productsCollection.updateOne({PRODUCT_ID: pID}, {
      $push: {REVIEWS: review} //bagam review ul in array ul REVIEWS
    })

  }

// await productsCollection.updateMany({}, { //aici avem pt a sterge pt a putea demonstra ca functioneaza
//   $unset: {REVIEWS: ""}
// })

} catch (err) {
  console.log(err.stack);
}

client.close();
}

main().catch(console.error);

```

## Join în MongoDB între user\_address, countries și regions

MongoDB Compass - localhost:27017/projectDB.user\_address

Connect Edit View Collection Help

localhost:27017 Documents projectDB.user\_a...

My Queries Databases Search

admin config local projectDB

- countries
- genres
- orders
- product\_from\_order
- product\_genres
- product\_type
- products
- publishers
- regions
- reviews
- user\_address**
- user\_payment
- users

projectDB.user\_address 33 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find More Options

ADD DATA EXPORT DATA 1 - 20 of 33

```

_id: ObjectId('647d7b8eda1853984cb3dcb6')
USER_ADDRESS_ID: 2
USER_ID: 2
state/province: "California"
CITY: "Los Angeles"
STREET: "Main Street"
ZIP: "90001"
PHONE: "555-123-4567"
COUNTRY: "United States of America"
REGION: "North America"

```

```

_id: ObjectId('647d7b8eda1853984cb3dcb7')
USER_ADDRESS_ID: 3
USER_ID: 3
state/province: "Ontario"
CITY: "Toronto"
STREET: "First Avenue"
ZIP: "M5V 1J2"
PHONE: "416-789-1234"
COUNTRY: "Canada"
REGION: "North America"

```

```
const { MongoClient } = require('mongodb');
```

```
async function main() {
```

```
  const url = "mongodb://localhost:27017";
```



```
const client = new MongoClient(url, { useUnifiedTopology: true });

try {
  await client.connect();
  console.log('Connected successfully to server');

  const db = client.db('proiectDB');
  const countriesCollection = db.collection('countries');
  const regionsCollection = db.collection('regions');
  const userAddressCollection = db.collection('user_address');

  const countries = await countriesCollection.find().toArray();
  const regions = await regionsCollection.find().toArray();
  const userAddress = await userAddressCollection.find().toArray();

  for(let region of regions){
    await countriesCollection.updateMany({REGION_ID: region.REGION_ID}, {
      $set: {REGION: region.NAME}
    })
  }

  await countriesCollection.updateMany({}, {
    $unset: {REGION_ID: ""}
  })

  for(let country of countries){
    await userAddressCollection.updateMany({COUNTRY_ID: country.COUNTRY_ID}, {
      $set: {COUNTRY: country.NAME,
        REGION: country.REGION}
    })
  }

  await userAddressCollection.updateMany({}, {
    $unset: {COUNTRY_ID: ""}
  })

} catch (err) {
  console.log(err.stack);
}

client.close();
}

main().catch(console.error);
```

Tranzacții: ilustrarea consistency levels in Oracle cu tranzații care operează asupra modelului ales.

Dirty write

```
-- T1

CREATE TABLE PRD AS SELECT * FROM PRODUCTS;

--EXEMPLU DE DIRTY WRITE:

--T1

SELECT PRICE FROM PRD WHERE PRODUCT_ID = 3;

-- 35

UPDATE PRD
SET PRICE = PRICE - 5
WHERE PRODUCT_ID = 3
AND PRICE > 10;

-- 30

-- T2
UPDATE PRD
SET PRICE = PRICE - 10
WHERE PRODUCT_ID = 3
AND PRICE > 10;
-- aici ar fi 20

--dar face T1 rollback inainte sa dea T2 commit
-- T1
rollback;
-- A REVENIT LA 35

-- T2
SELECT PRICE FROM PRD WHERE PRODUCT_ID = 3;

-- VA FI 25 LA AMANDOI
commit;

-- T1
SELECT PRICE FROM PRD WHERE PRODUCT_ID = 3;
```

```
-- 25
```

Lost update

```
--EXEMPLU DE LOST UPDATE:
```

```
-- T1
```

```
SELECT PRICE FROM PRD WHERE PRODUCT_ID = 3;
```

```
-- 35
```

```
UPDATE PRD
```

```
SET PRICE = PRICE + 10
```

```
WHERE PRODUCT_ID = 3;
```

```
-- 45
```

```
-- T2
```

```
UPDATE PRD
```

```
SET PRICE = PRICE + 20
```

```
WHERE PRODUCT_ID = 3;
```

```
-- 65
```

```
-- T1
```

```
commit
```

```
-- T2
```

```
SELECT SALARY FROM PRD WHERE PRODUCT_ID = 3;
```

```
commit
```

```
-- T1
```

```
SELECT SALARY FROM PRD WHERE PRODUCT_ID = 3;
```

```
-- DIFERENTA DINTRE DIRTY WRITE I LOST UPDATE ESTE FAPTUL CA T1 FACE COMMIT IN LOC  
DE ROLLBACK
```