

MySQL

.Partea IV. Limbajul SQL (II)

Ce ne așteaptă?

1. Instrucțiunea INSERT INTO
2. Instrucțiunea SELECT + FROM
3. Instrucțiunea SELECT + ORDER BY
4. Instrucțiunea SELECT + WHERE
5. Instrucțiunea SELECT + JOIN
6. Instrucțiunea UPDATE
7. Instrucțiunea DELETE

1. Instrucțiunea INSERT INTO

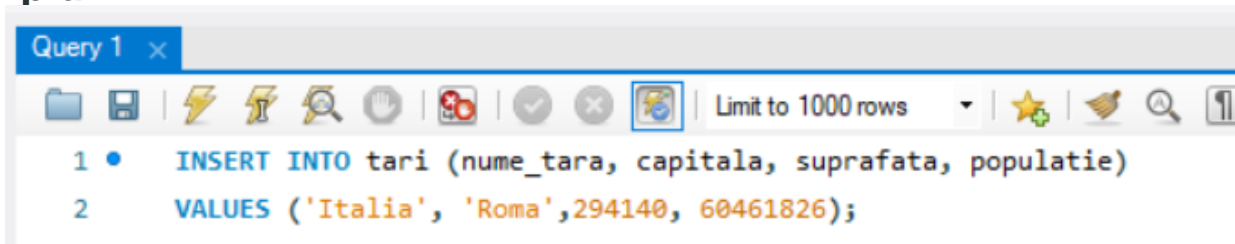
Introducerea datelor în tabel

- Instrucțiunea INSERT INTO – inscrierea datelor in tabele

- Inscrierea datelor in tabel

```
INSERT INTO nume_tabel (nume_col_1, nume_col_2, nume_col_3, nume_col_4)  
VALUES (valoarea_col_1, valoarea_col_2, valoare_col_3, valoarea_col_4);
```

- În mod obligatoriu se vor specifica valorile pentru coloanele ce nu acceptă valori nule (Not Null) și nu au alt mecanism de generare a acestora (Auto Increment, Defaul/Expression, etc)
- Exemplu



The screenshot shows a SQL query editor window titled "Query 1". The toolbar includes icons for file operations, execution, and search. The query text is as follows:

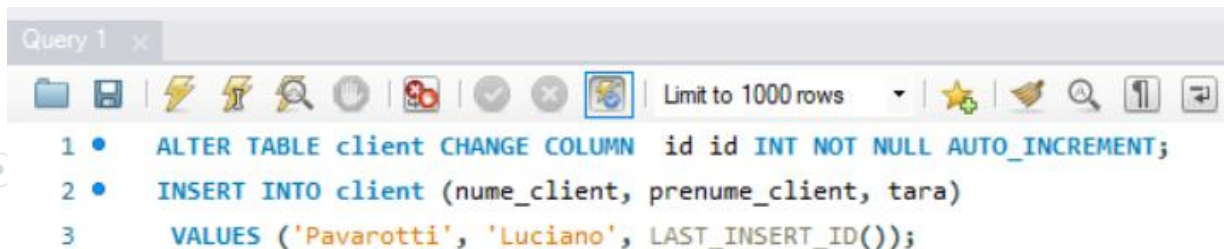
```
1 • INSERT INTO tari (nume_tara, capitala, suprafata, populatie)  
2   VALUES ('Italia', 'Roma', 294140, 60461826);  
-
```

Introducerea datelor în tabel cu chei străine

- Pentru a introduce date într-un tabel cu cheie străină este necesară cunoașterea valorii în coloana de legătură
- Pentru determinarea valorii cheii primare a ultimei introduceri se utilizează funcția `LAST_INSERT_ID()`
- Introducerea datelor în tabel cu cheie străină

```
INSERT INTO nume_tabel (nume_col_1, nume_col_2, nume_col_3_cheie_str)  
VALUES (valoarea_col_1, valoarea_col_2, LAST_INSERT_ID());
```

- Exemplu (inițial se setează coloana id să fie cu Auto Increment)



```
Query 1 x  
1 • ALTER TABLE client CHANGE COLUMN id id INT NOT NULL AUTO_INCREMENT;  
2 • INSERT INTO client (nume_client, prenume_client, tara)  
3   VALUES ('Pavarotti', 'Luciano', LAST_INSERT_ID());
```

2. Instrucțiunea SELECT + FROM

Instrucțiunea SELECT

- Instrucțiunea SELECT – selectarea și prezentarea variabilelor, rezultatelor operațiilor și a datelor din baza de date
- În cazul utilizării instrucțiunii SELECT pentru prezentarea datelor din bază, aceasta se utilizează cu alte cuvinte cheie și anume:
 - FROM – specifica tabelul asupra căruia se aplica SELECT
 - ORDER BY – permite ordonarea rezultatelor lui SELECT
 - WHERE – permite aplicarea unor condiții de selectare cu SELECT
 - (INNER, LEFT, RIGHT) JOIN – permite interacțiunea dintre 2 tabele SELECT

Citirea datelor unui tabel

- **FROM** – permite specificarea tabelului bazei de date
- **Prezentarea tuturor datelor unui tabel din baza de date**

```
SELECT * FROM nume_tabel;
```

- **Exemplu (se mai introduc preventiv mai multe date în tabele)**

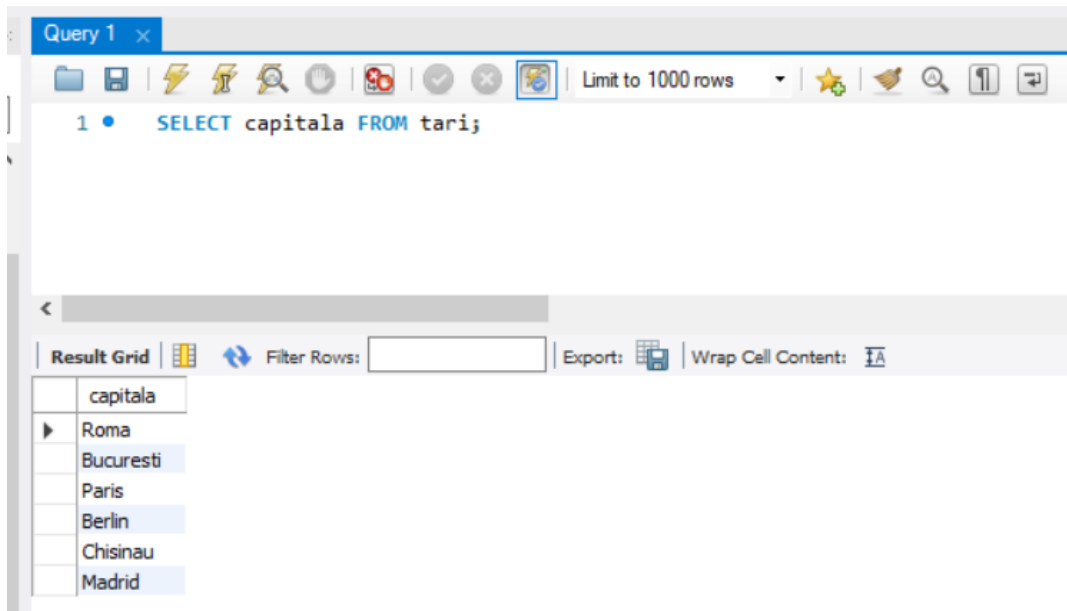
Query 1

Citirea datelor unei coloane a tabelului

- **Prezentarea datelor unei coloane a unui tabel din baza de date**

```
SELECT nume_coloana FROM nume_tabel;
```

- **Exemplu**



The screenshot shows a SQL query editor window titled "Query 1". The query text is `SELECT capitala FROM tari;`. The results are displayed in a table with the following data:

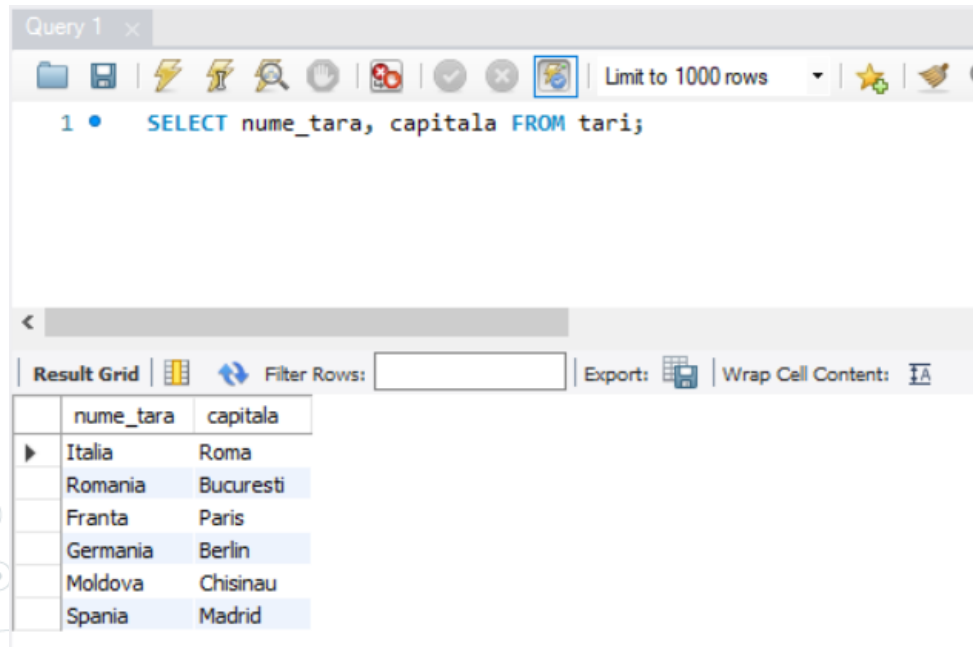
capitala
Roma
Bucuresti
Paris
Berlin
Chisinau
Madrid

Citirea datelor mai multor coloane a tabelului

- Prezentarea datelor câtorva coloane a unui tabel din baza de date

```
SELECT nume_coloana1, nume_coloana1 FROM nume_tabel;
```

- Exemplu



The screenshot shows a database query tool interface. At the top, a tab labeled 'Query 1' is active. Below it is a toolbar with various icons, including a 'Limit to 1000 rows' dropdown. The SQL query entered is: `1 • SELECT nume_tara, capitala FROM tari;`. Below the query editor, there is a 'Result Grid' section. It includes a 'Filter Rows' input field and an 'Export' button. The results are displayed in a table with two columns: 'nume_tara' and 'capitala'. The table contains six rows of data, with the first row highlighted by a mouse cursor.

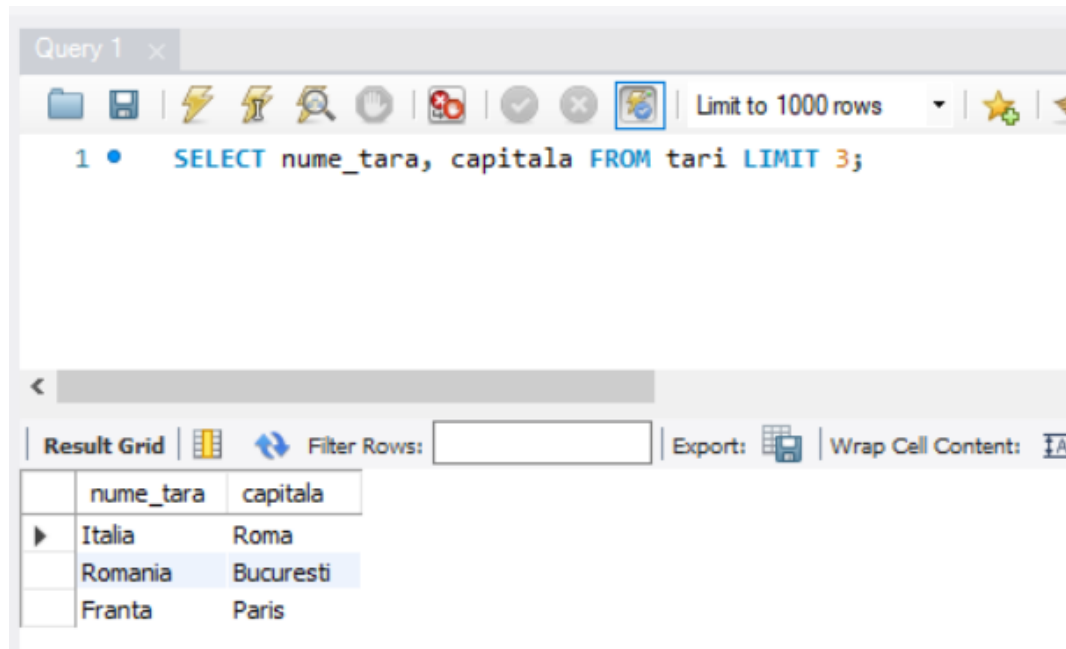
nume_tara	capitala
Italia	Roma
Romania	Bucuresti
Franta	Paris
Germania	Berlin
Moldova	Chisinau
Spania	Madrid

Citirea doar a primelor n date

- **LIMIT** - permite introducerea unor limitări a numărului de rezultate afișate
- **Limitarea numărului de rezultate prezentate la primele n**

```
SELECT nume_coloana1, nume_coloana2 FROM nume_tabel LIMIT n;
```

- **Exemplu**



The screenshot shows a database query editor window titled "Query 1". The query text is: `1 • SELECT nume_tara, capitala FROM tari LIMIT 3;`. The results are displayed in a table with two columns: "nume_tara" and "capitala". The first three rows of the table are visible: Italia (Roma), Romania (Bucuresti), and Franta (Paris). The "Romania" row is highlighted. The interface includes a toolbar with various icons and a "Limit to 1000 rows" dropdown menu.

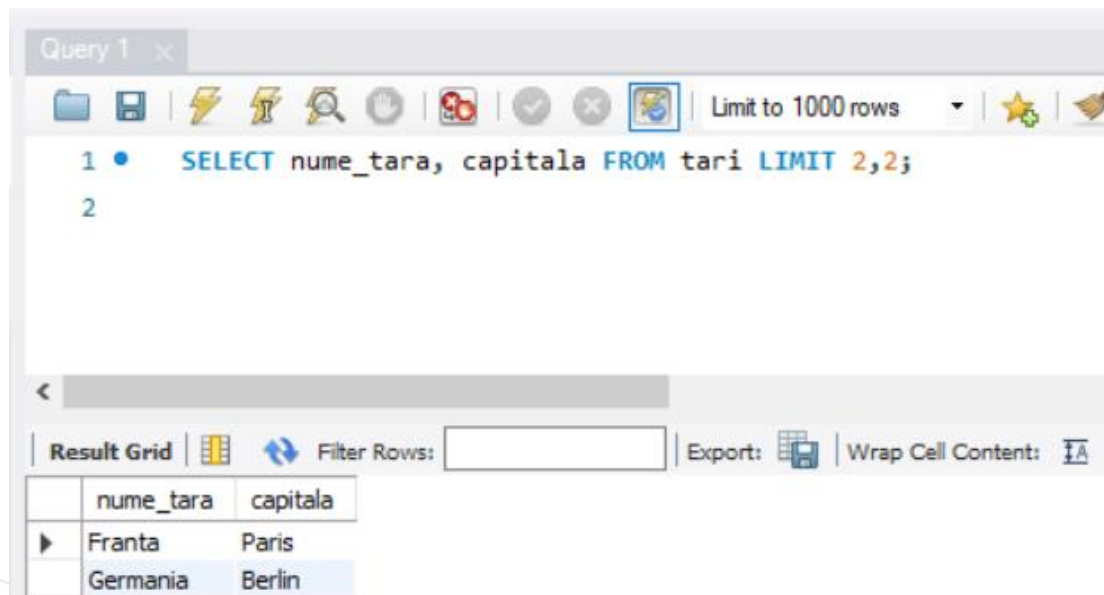
nume_tara	capitala
Italia	Roma
Romania	Bucuresti
Franta	Paris

Citirea doar a unui interval de date (1)

- Limitarea numărului de rezultate prezentate la valorile dintre m și $m+n$

```
SELECT nume_coloana1, nume_coloana2 FROM nume_tabel LIMIT m,n;
```

- Exemplu



The screenshot shows a database query editor window titled "Query 1". The query text is:

```
1 • SELECT nume_tara, capitala FROM tari LIMIT 2,2;
```

The results are displayed in a table with two columns: "nume_tara" and "capitala". The first two rows are highlighted:

nume_tara	capitala
Franta	Paris
Germania	Berlin

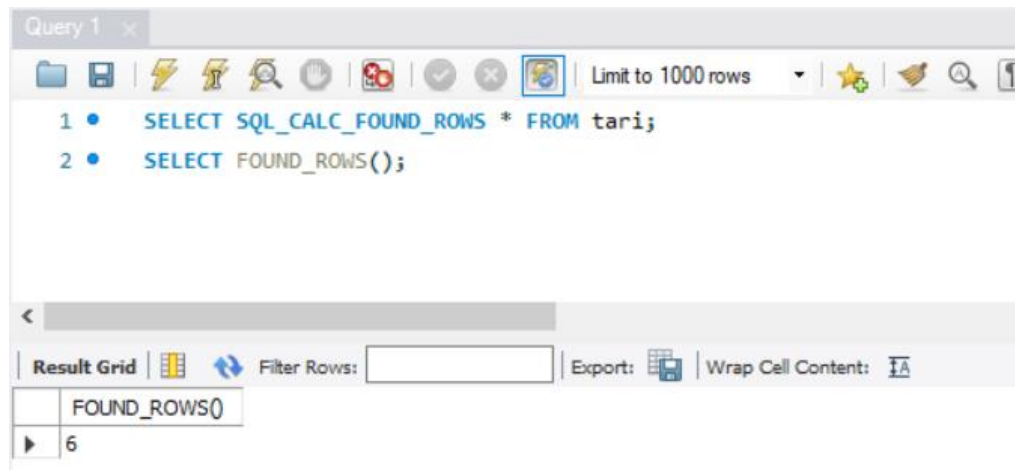
The interface includes a toolbar with various icons, a "Limit to 1000 rows" dropdown, and a "Result Grid" tab. The "Filter Rows" field is empty, and the "Export" button is visible.

Citirea numărului de înscrieri în tabel (1)

- Prezentarea numărului total de valori se realizează în două etape: calcularea numărului de valori cu ajutorul SQL_CALC_FOUND_ROWS și prezentarea cu ajutorul FOUND_ROWS()

```
SELECT SQL_CALC_FOUND_ROWS * FROM nume_tabel;  
SELECT FOUND_ROWS();
```

- Exemplu



Citirea numărului de înregistrări în tabel (2)

- Funcția **COUNT()** – determină numărul de valori într-o coloana
- Prezentarea numărului de înregistrări într-un tabel cu ajutorul funcției **COUNT()**

```
SELECT COUNT(ume_coloana_chei_primara) FROM ume_tabel;
```

- Exemplu

The screenshot shows a database query editor window. The title bar includes 'Query 1', 'client', 'client - Table', and 'client'. The toolbar contains various icons for file operations, execution, and viewing. The SQL editor displays the query: `1 SELECT COUNT(id_tara) FROM tari;`. The 'Result Grid' tab is active, showing a single row with the column header 'COUNT(id_tara)' and the value '6'. The 'Filter Rows' field is empty, and the 'Export' button is visible.

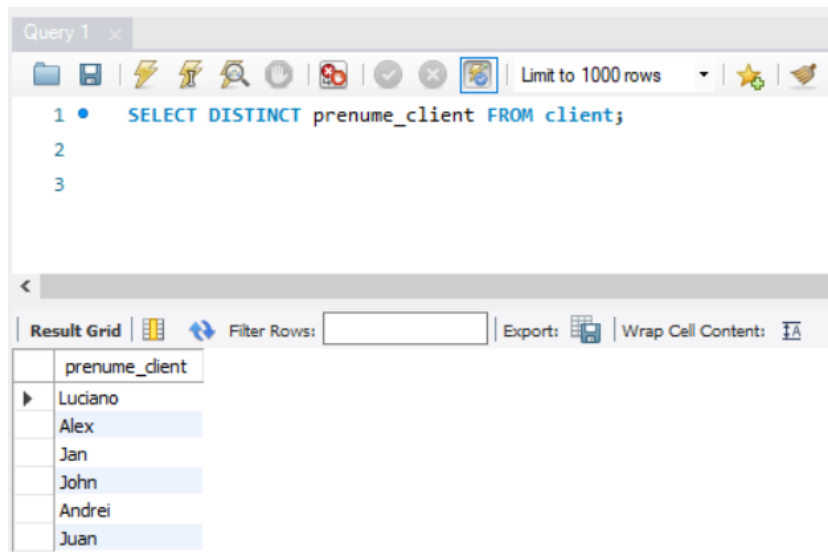
COUNT(id_tara)
6

Prezentarea valorilor unice pe coloana

- **DISTINCT** – permite detectarea valorilor unice într-o coloana
- **Prezentarea valorilor unice într-o coloana a tabelului**

```
SELECT DISTINCT nume_coloana FROM nume_tabel;
```

- **Exemplu (pentru explicitatea trebuie să existe mai mulți clienți cu un prenume)**



The screenshot shows a SQL query editor window titled "Query 1". The query text is:

```
1 • SELECT DISTINCT prenume_client FROM client;
```

The editor includes a toolbar with various icons and a "Limit to 1000 rows" dropdown. Below the query editor, there is a "Result Grid" section. The grid has a single column labeled "prenume_client" and displays the following values:

prenume_client
Luciano
Alex
Jan
John
Andrei
Juan

Prezentarea numărului de valori unice pe coloană

- Prezentarea numărului de valori unice într-o coloană a tabelului

```
SELECT COUNT(DISTINCT nume_coloana) FROM nume_tabel;
```

- Exemplu

The screenshot shows a SQL query editor window titled "Query 1". The query text is:

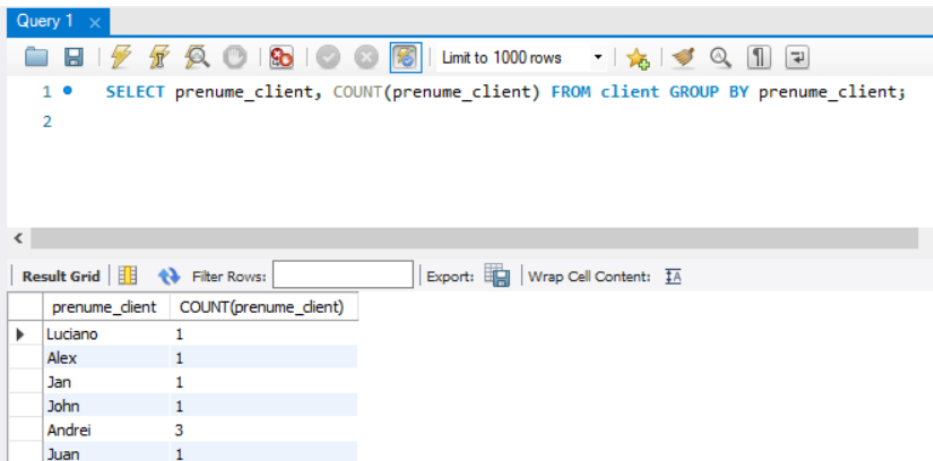
```
1 • SELECT COUNT(DISTINCT prenume_client) FROM client;
```

Below the query editor, the "Result Grid" is displayed. It shows a single column with the header "COUNT(DISTINCT prenume_client)" and a single row with the value "6".

COUNT(DISTINCT prenume_client)
6

Gruparea datelor după valorile unei coloane

- **GROUP BY** – permite gruparea datelor după valorile unei coloane
- Pentru a se diferenția de **DISTINCT**, **GROUP BY** utilizează și funcții de agregare (**COUNT()**, **MAX()**, **MIN()**, **SUM()**, **AVR()**, etc)
- Gruparea datelor după valoarea unei coloane si afișarea numărului fiecărui grup
`SELECT nume_coloana, COUNT(nume_coloana) FROM nume_tabel GROUP BY nume_coloana`
- Exemplu: afișarea tuturor grupurilor din coloana `nume_client` a tabelului `client` și a numărului de date in fiecare grup



The screenshot shows a database query editor window titled "Query 1". The SQL query entered is: `SELECT prenume_client, COUNT(prenume_client) FROM client GROUP BY prenume_client;`. Below the query editor, the results are displayed in a table with two columns: `prenume_client` and `COUNT(prenume_client)`. The results show the following data:

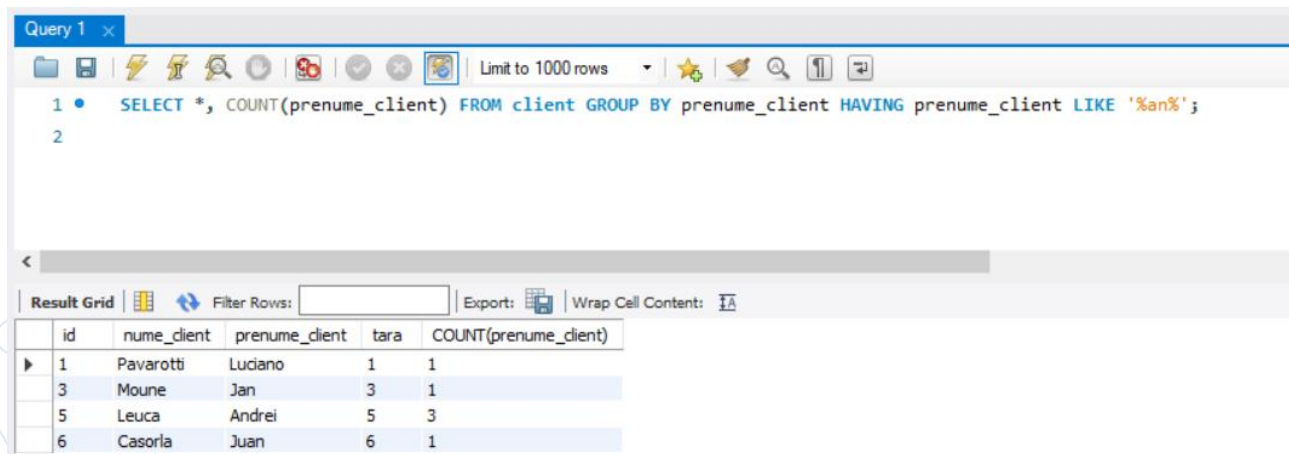
prenume_client	COUNT(prenume_client)
Luciano	1
Alex	1
Jan	1
John	1
Andrei	3
Juan	1

Filtrarea grupurilor de date

- **HAVING** – permite introducerea unei condiții la prezentarea rezultatelor grupării datelor după valorile unei coloane
- Gruparea datelor după valoarea unei coloane și afișarea doar a grupurilor ce respectă o condiție

`SELECT nume_coloane FROM nume_tabel GROUP BY nume_coloana HAVING conditie`

- **Exemplu:** afișarea datelor grupurilor formate conform coloanei `nume_client` care a un secvența an în interior și a numărului de date în fiecare grup



The screenshot shows a database query editor window titled "Query 1". The SQL query entered is: `SELECT *, COUNT(prenume_client) FROM client GROUP BY prenume_client HAVING prenume_client LIKE '%an%';`. The results are displayed in a table with 5 columns: `id`, `nume_client`, `prenume_client`, `tara`, and `COUNT(prenume_client)`. The results table contains 5 rows of data.

id	nume_client	prenume_client	tara	COUNT(prenume_client)
1	Pavarotti	Ludano	1	1
3	Moune	Jan	3	1
5	Leuca	Andrei	5	3
6	Casorla	Juan	6	1

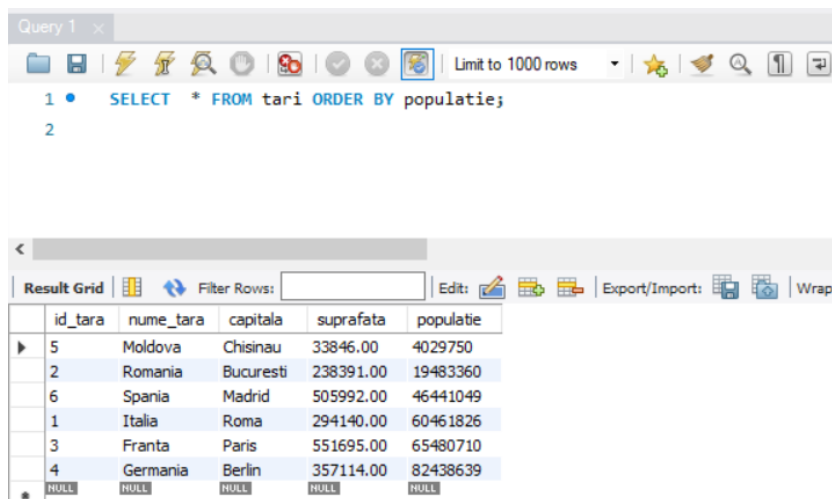
3. Instrucțiunea SELECT + ORDER BY

Aranjarea în ordine crescătoare

- **ORDER BY** – asigură ordonarea datelor în ordine crescătoare (ASC - implicit) sau descrescătoare (DESC)
- **Prezentarea datelor ordonate crescător după valorile unei coloane**

```
SELECT * FROM nume_tabel ORDER BY nume_coloana;
```

Exemplu



The screenshot shows a database query editor with a query window and a result grid. The query window displays the following SQL query:

```
1 • SELECT * FROM tari ORDER BY populatie;  
2
```

The result grid shows the following data:

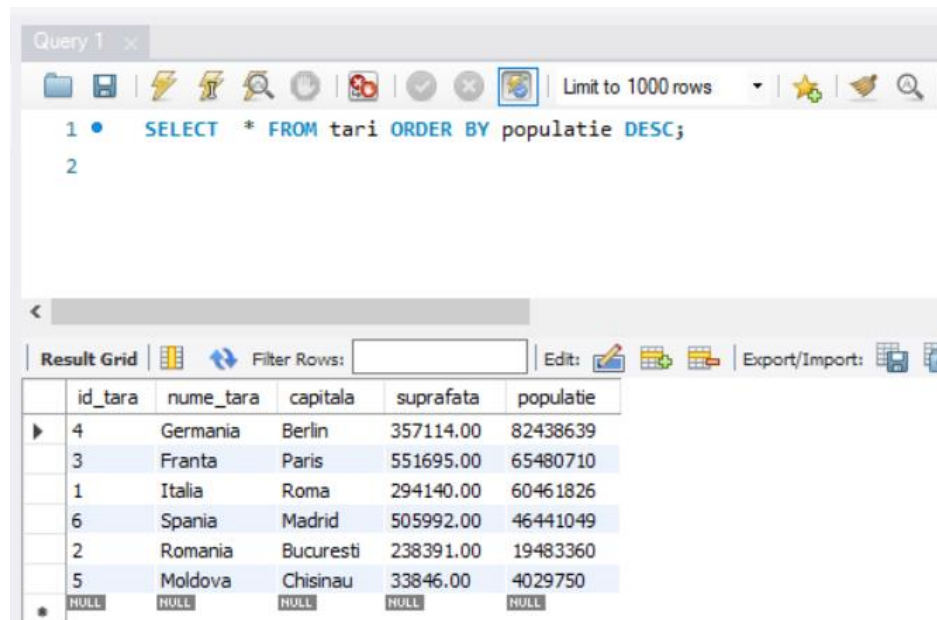
	id_tara	nume_tara	capitala	suprafata	populatie
5		Moldova	Chisinau	33846.00	4029750
2		Romania	Bucuresti	238391.00	19483360
6		Spania	Madrid	505992.00	46441049
1		Italia	Roma	294140.00	60461826
3		Franta	Paris	551695.00	65480710
4		Germania	Berlin	357114.00	82438639
*		NULL	NULL	NULL	NULL

Aranjarea în ordine descrescătoare

- Prezentarea datelor ordonate decrescător după valorile unei coloane

```
SELECT * FROM nume_tabel ORDER BY nume_coloana DESC;
```

- Exemplu



The screenshot shows a database query editor window titled "Query 1". The query entered is: `SELECT * FROM tari ORDER BY populatie DESC;`. Below the query editor, the "Result Grid" displays the results of the query. The results are ordered by population in descending order. The table has columns: id_tara, nume_tara, capitala, suprafata, and populatie. The data rows are as follows:

	id_tara	nume_tara	capitala	suprafata	populatie
▶	4	Germania	Berlin	357114.00	82438639
	3	Franta	Paris	551695.00	65480710
	1	Italia	Roma	294140.00	60461826
	6	Spania	Madrid	505992.00	46441049
	2	Romania	Bucuresti	238391.00	19483360
	5	Moldova	Chisinau	33846.00	4029750
*	NULL	NULL	NULL	NULL	NULL

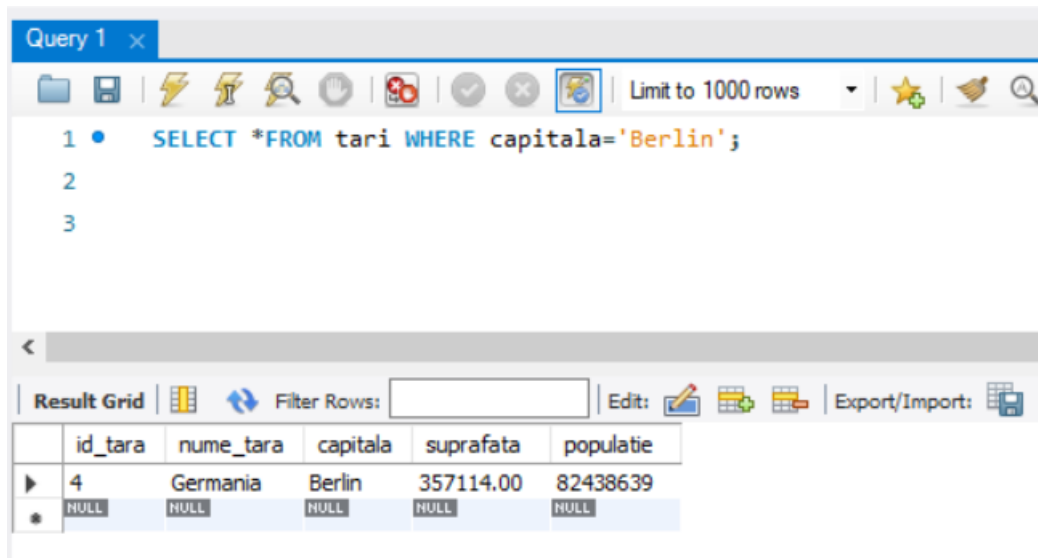
4. Instrucțiunea SELECT + WHERE

Citirea datelor conform unei valori a coloanei

- **WHERE** – permite introducerea unei condiții de selecție a datelor
- **Prezentarea tuturor datelor care au o anumita valoare într-o anumita coloana**

`SELECT *FROM nume_tabel WHERE nume_coloana=valoare coloana;`

- **Exemplu**



The screenshot shows a SQL query editor window titled "Query 1". The query text is: `1 • SELECT *FROM tari WHERE capitala='Berlin';`. Below the query, there are three empty lines numbered 2 and 3. The results are displayed in a "Result Grid" at the bottom. The grid has five columns: `id_tara`, `nume_tara`, `capitala`, `suprafata`, and `populatie`. The first row of data shows Germany with `id_tara` 4, `nume_tara` Germania, `capitala` Berlin, `suprafata` 357114.00, and `populatie` 82438639. Below this row, there is a row of NULL values. The interface includes a toolbar with various icons and a "Limit to 1000 rows" dropdown.

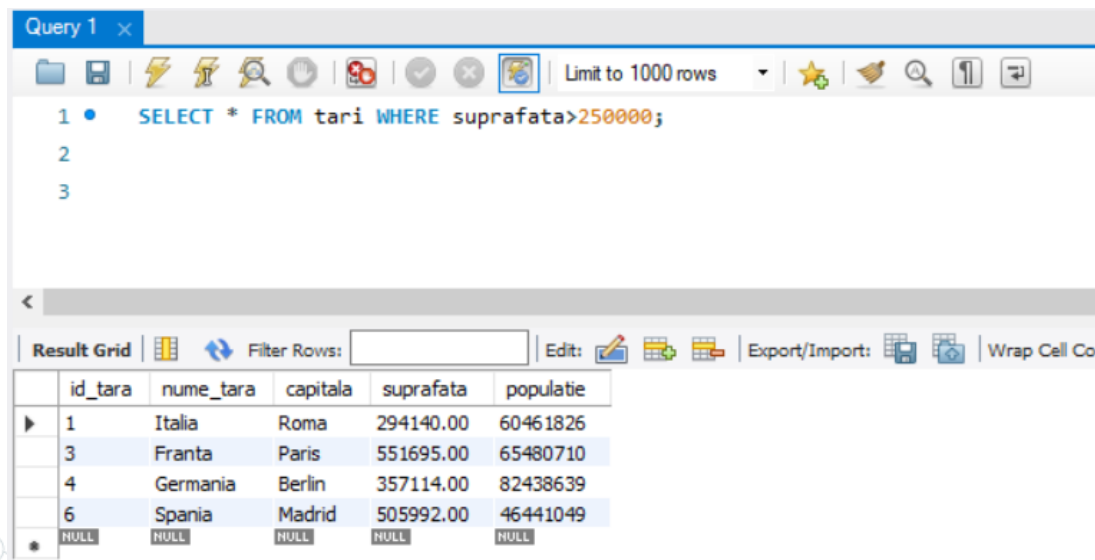
	id_tara	nume_tara	capitala	suprafata	populatie
▶	4	Germania	Berlin	357114.00	82438639
*	NULL	NULL	NULL	NULL	NULL

Citirea datelor conform unei condiții a coloanei

- Prezentarea tuturor datelor pentru care valoarea dintr-o anumita coloana îndeplinește o condiție

```
SELECT *FROM nume_tabel WHERE nume_coloana>valoare;
```

- Exemplu



The screenshot shows a database query tool interface. At the top, a tab labeled 'Query 1' is active. Below the tab is a toolbar with various icons for file operations, execution, and viewing. The SQL query editor contains the following query:

```
1 • SELECT * FROM tari WHERE suprafata>250000;  
2  
3
```

Below the query editor is a 'Result Grid' section. It includes a 'Filter Rows' input field and an 'Edit' button. The result grid displays the following data:

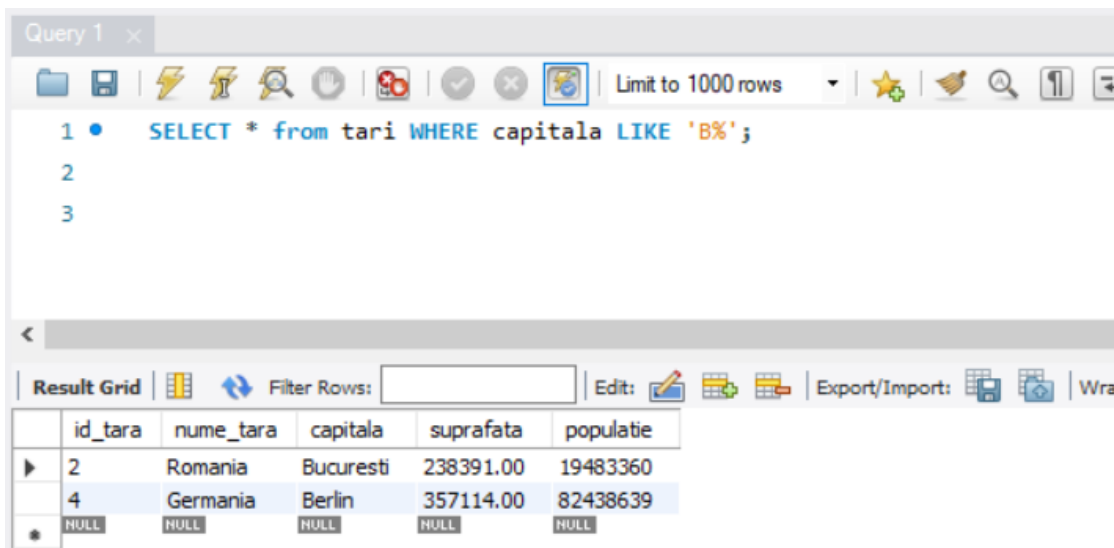
	id_tara	nume_tara	capitala	suprafata	populatie
▶	1	Italia	Roma	294140.00	60461826
	3	Franta	Paris	551695.00	65480710
	4	Germania	Berlin	357114.00	82438639
	6	Spania	Madrid	505992.00	46441049
*	NULL	NULL	NULL	NULL	NULL

Citirea datelor conform începutului unei date

- **LIKE** – permite detecția prezenței unor secvențe în datele unei coloane
- **Prezentarea tuturor datelor pentru care valoarea dintr-o anumita coloana începe cu o anumita secvență**

```
SELECT *FROM nume_tabel WHERE nume_coloana LIKE 'secventa%';
```

- **Exemplu: Prezentarea tuturor rezultatelor din tabelul tari pentru care capitala începe cu litera B**



The screenshot shows a database query tool interface. The query editor displays the following SQL query:

```
1 • SELECT * from tari WHERE capitala LIKE 'B%';
```

The results are shown in a table with the following columns: id_tara, nume_tara, capitala, suprafata, and populatie. The results table contains three rows:

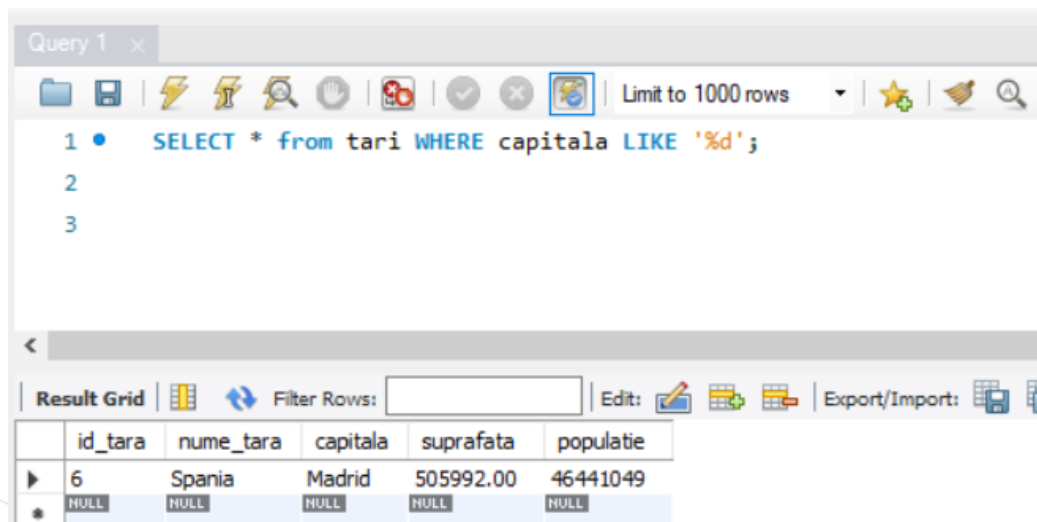
	id_tara	nume_tara	capitala	suprafata	populatie
▶	2	Romania	Bucuresti	238391.00	19483360
	4	Germania	Berlin	357114.00	82438639
•	NULL	NULL	NULL	NULL	NULL

Citirea datelor conform sfârșitului unei date

- Prezentarea tuturor datelor pentru care valoarea dintr-o anumita coloana se termină cu o anumita secvență

```
SELECT *FROM nume_tabel WHERE nume_coloana LIKE '%secventa';
```

- Exemplu: Prezentarea tuturor rezultatelor din tabelul tari pentru care capitala se termină cu litera d



The screenshot shows a database query editor window titled 'Query 1'. The SQL query entered is: `SELECT * from tari WHERE capitala LIKE '%d';`. The results are displayed in a 'Result Grid' table with 5 columns: `id_tara`, `nume_tara`, `capitala`, `suprafata`, and `populatie`. The first row shows data for Spain (id_tara: 6, nume_tara: Spania, capitala: Madrid, suprafata: 505992.00, populatie: 46441049). A second row is partially visible with NULL values.

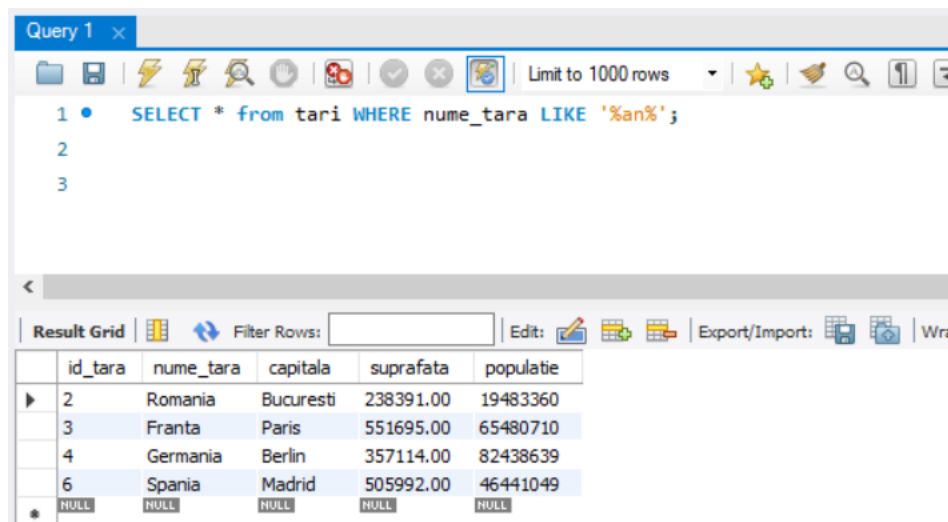
	id_tara	nume_tara	capitala	suprafata	populatie
▶	6	Spania	Madrid	505992.00	46441049
*	NULL	NULL	NULL	NULL	NULL

Citirea datelor conform conținutului unei date

- **Prezentarea tuturor datelor pentru care valoarea dintr-o anumita coloana conține o anumita secvență**

```
SELECT *FROM nume_tabel WHERE nume_coloana LIKE '%secventa%';
```

- **Exemplu: Prezentarea tuturor rezultatelor din tabelul tari pentru care numele țării conține secvența an**



The screenshot shows a SQL query editor window titled 'Query 1'. The query is: `SELECT * from tari WHERE nume_tara LIKE 'an%';`. Below the query, the 'Result Grid' displays the results of the query. The table has columns: id_tara, nume_tara, capitala, suprafata, and populatie. The results show four rows of data for countries starting with 'an': Romania, Franta, Germania, and Spania. The last row is a summary row with NULL values.

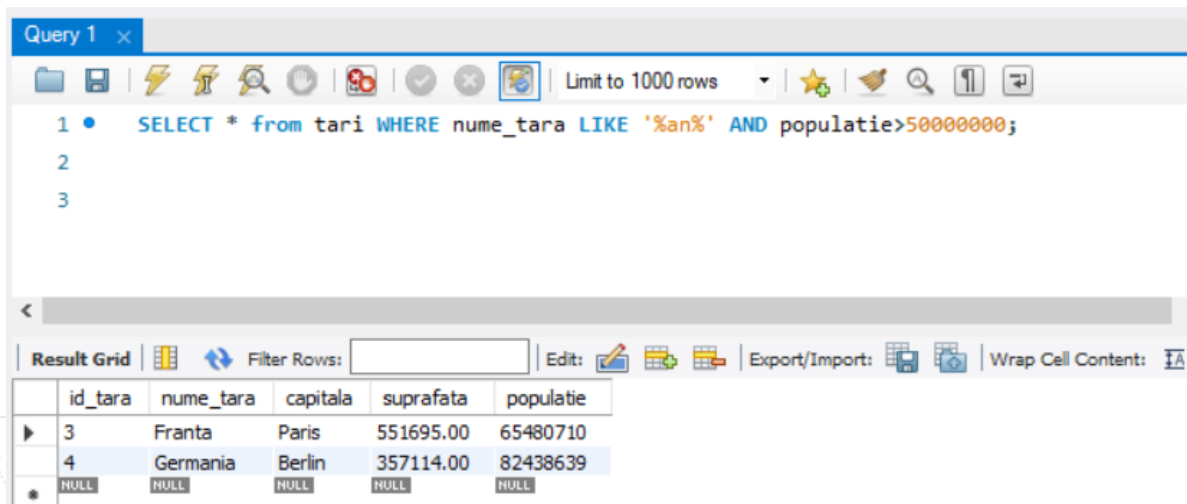
	id_tara	nume_tara	capitala	suprafata	populatie
▶	2	Romania	Bucuresti	238391.00	19483360
	3	Franta	Paris	551695.00	65480710
	4	Germania	Berlin	357114.00	82438639
	6	Spania	Madrid	505992.00	46441049
*	NULL	NULL	NULL	NULL	NULL

Citirea datelor conform condițiilor multiple

- Prezentarea tuturor datelor pentru care se respecta conditii pe mai multe coloane

```
SELECT *FROM nume_tabel WHERE conditie_colona1 AND conditie_coloana2;
```

- Exemplu: Prezentarea tuturor rezultatelor din tabelul tari pentru care numele țării conține secvența an iar populație este mai marea de 50 milioane



The screenshot shows a database query tool interface. At the top, a tab labeled 'Query 1' is active. Below it is a toolbar with various icons for file operations, execution, and viewing. The SQL query editor contains the following query:

```
1 • SELECT * from tari WHERE nume_tara LIKE '%an%' AND populatie>50000000;  
2  
3
```

Below the query editor is a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Edit' button, and 'Export/Import' and 'Wrap Cell Content' options. The result grid displays the following data:

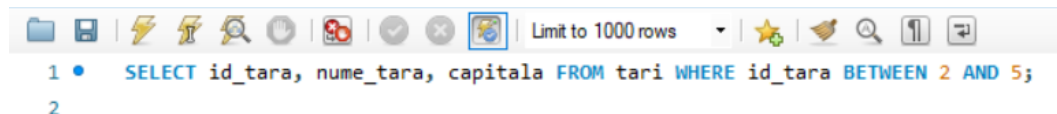
	id_tara	nume_tara	capitala	suprafata	populatie
▶	3	Franta	Paris	551695.00	65480710
	4	Germania	Berlin	357114.00	82438639
•	NULL	NULL	NULL	NULL	NULL

Citirea doar a unui interval de date (2)

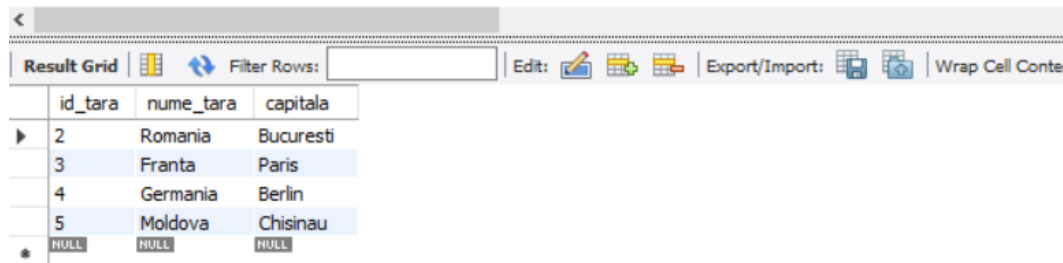
- **BETWEEN** – permite specificarea intervalului de valori a uni coloane
- Prezentarea tuturor datelor o anumită coloana ia valori într-un interval specificat

```
SELECT *FROM nume_tabel WHERE nume_col BETWEEN val_initial AND val_final;
```

- **Exemplu:**



The screenshot shows a toolbar with various icons for file operations, execution, and editing. Below the toolbar, the SQL query is displayed: `1 • SELECT id_tara, nume_tara, capitala FROM tari WHERE id_tara BETWEEN 2 AND 5;` followed by a line number `2`.



The screenshot shows a database application interface with a 'Result Grid' tab. The grid displays the results of the SQL query, showing columns 'id_tara', 'nume_tara', and 'capitala'. The rows are numbered 2 through 5, corresponding to the 'BETWEEN' clause in the query. Row 2 is highlighted. Below the grid, there are buttons for 'Edit', 'Export/Import', and 'Wrap Cell Conte'.

	id_tara	nume_tara	capitala
2		Romania	Bucuresti
3		Franta	Paris
4		Germania	Berlin
5		Moldova	Chisinau
*	NULL	NULL	NULL

5. Instrucțiunea SELECT + JOIN

Cuvântul cheie JOIN

- **JOIN** – permite asocierea datelor din mai multe coloane a diferitor tabele relaționale
- **ON** – stabilește criteriul de asociere prin compararea datelor unei coloane a unui tabel cu datele unei coloane a altui tabel
- Pentru excluderea posibilelor erori de coincidență, dar și pentru o vizibilitate mai bună, numele coloanelor vor avea ca prefix numele tabelului
- În funcție de mecanismul includerii ce nu respecta criteriul de asociere pot fi:
 - **INNER JOIN** – include doar datele din ambele tabele ce respectă criteriul de asociere
 - **LEFT JOIN** – include toate datele din tabelul stâng iar din cel drept doar datele ce respectă criteriu
 - **RIGHT JOIN** – include toate datele din tabelul drept și iar din cel stâng doar datele ce respectă criteriul

Crearea tabelelor pentru exemplificare

- Se creează un tabel a cu setările și datele

Table Name: Schema: **magazin_online**

Charset/Collation: Engine: **InnoDB**

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
<input checked="" type="checkbox"/> id_a	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
<input checked="" type="checkbox"/> nume_a	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

1 • **SELECT * FROM a;**

Result Grid | Filter Rows: | Export:

	id_a	nume_a
▶ 1		Europa
2		Asia
3		America
4		Africa

- Se creează un tabel b cu setările și datele

Table Name: Schema: **magazin_online**

Charset/Collation: Engine: **InnoDB**

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expr
<input checked="" type="checkbox"/> id_b	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
<input checked="" type="checkbox"/> nume_b	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

1 • **SELECT * FROM b;**

Result Grid | Filter Rows: | Export:

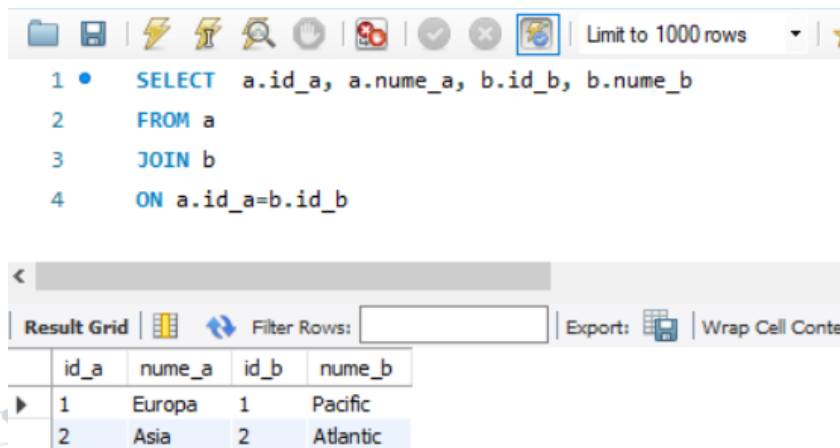
	id_b	nume_b
▶ 1		Pacific
2		Atlantic
5		Indian
6		Artic

Prezentarea asocierii INNER JOIN

- Prezentarea datelor din 2 tabele asociate după egalitatea valorile a două coloane si excluderea datelor ce nu respectă criteriul de asociere

```
SELECT nume_tabel_stânga.nume_coloane, nume_tabel_dreapta.nume_coloane  
FROM nume_tabel_stânga  
JOIN nume_tabel_dreapta  
ON nume_tabel_dreapta.nume_coloana_asociere=  
nume_tabel_stânga.nume_coloana_asociere;
```

- Exemplu Asocierea tabelului a cu b după criteriul id_a=id_b



The screenshot displays a SQL query editor with a toolbar at the top. The query text is as follows:

```
1 • SELECT a.id_a, a.nume_a, b.id_b, b.nume_b  
2 FROM a  
3 JOIN b  
4 ON a.id_a=b.id_b
```

Below the query editor, the 'Result Grid' is shown, displaying the results of the query. The grid has four columns: id_a, nume_a, id_b, and nume_b. It contains two rows of data:

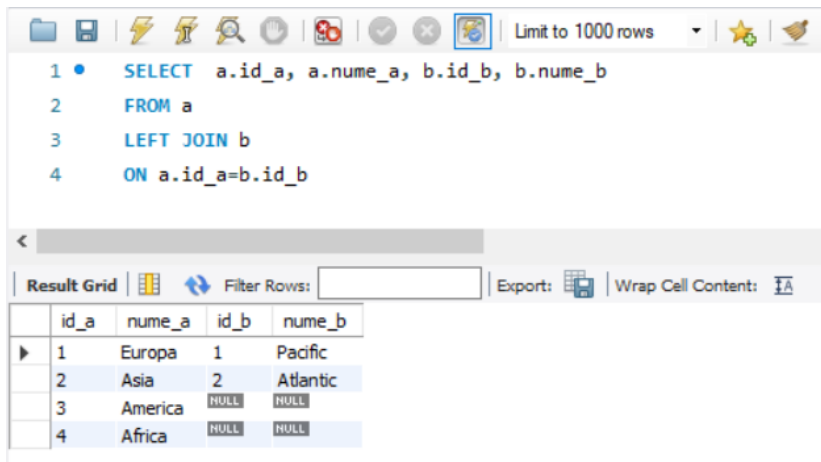
id_a	nume_a	id_b	nume_b
1	Europa	1	Pacific
2	Asia	2	Atlantic

Prezentarea asocierii LEFT JOIN

- **Prezentarea datelor din 2 tabele asociate după egalitatea valorile a două coloane si excluderea datelor ce nu respectă criteriul de asociere în tabelul drept (fața de JOIN)**

```
SELECT nume_tabel_stînga.nume_coloane, nume_tabel_dreapta.nume_coloane  
FROM nume_tabel_stînga  
LEFT JOIN nume_tabel_dreapta  
ON nume_tabel_dreapta.nume_coloana_asociere=  
nume_tabel_stînga.nume_coloana_asociere;
```

- **Exemplu Asocierea tabelului a cu b după criteriul id_a=id_b**



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • SELECT a.id_a, a.nume_a, b.id_b, b.nume_b  
2 FROM a  
3 LEFT JOIN b  
4 ON a.id_a=b.id_b
```

Below the query, the 'Result Grid' is displayed with the following data:

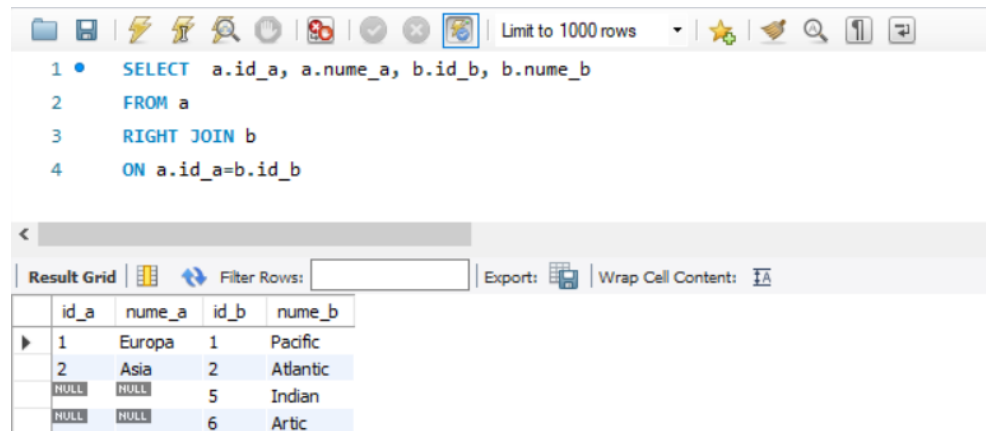
	id_a	nume_a	id_b	nume_b
1	1	Europa	1	Pacific
2	2	Asia	2	Atlantic
3	3	America	NULL	NULL
4	4	Africa	NULL	NULL

Prezentarea asocierii RIGHT JOIN

- Prezentarea datelor din 2 tabele asociate după egalitatea valorile a două coloane și excluderea datelor ce nu respectă criteriul de asociere în tabelul stâng (fața de JOIN)

```
SELECT nume_tabel_stânga.nume_coloane, nume_tabel_dreapta.nume_coloane  
FROM nume_tabel_stânga  
RIGHT JOIN nume_tabel_dreapta  
ON nume_tabel_dreapta.nume_coloana_asociere=  
nume_tabel_stânga.nume_coloana_asociere;
```

- Exemplu Asocierea tabelului a cu b după criteriul id_a=id_b



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • SELECT a.id_a, a.nume_a, b.id_b, b.nume_b  
2 FROM a  
3 RIGHT JOIN b  
4 ON a.id_a=b.id_b
```

Below the query, the 'Result Grid' is displayed with the following data:

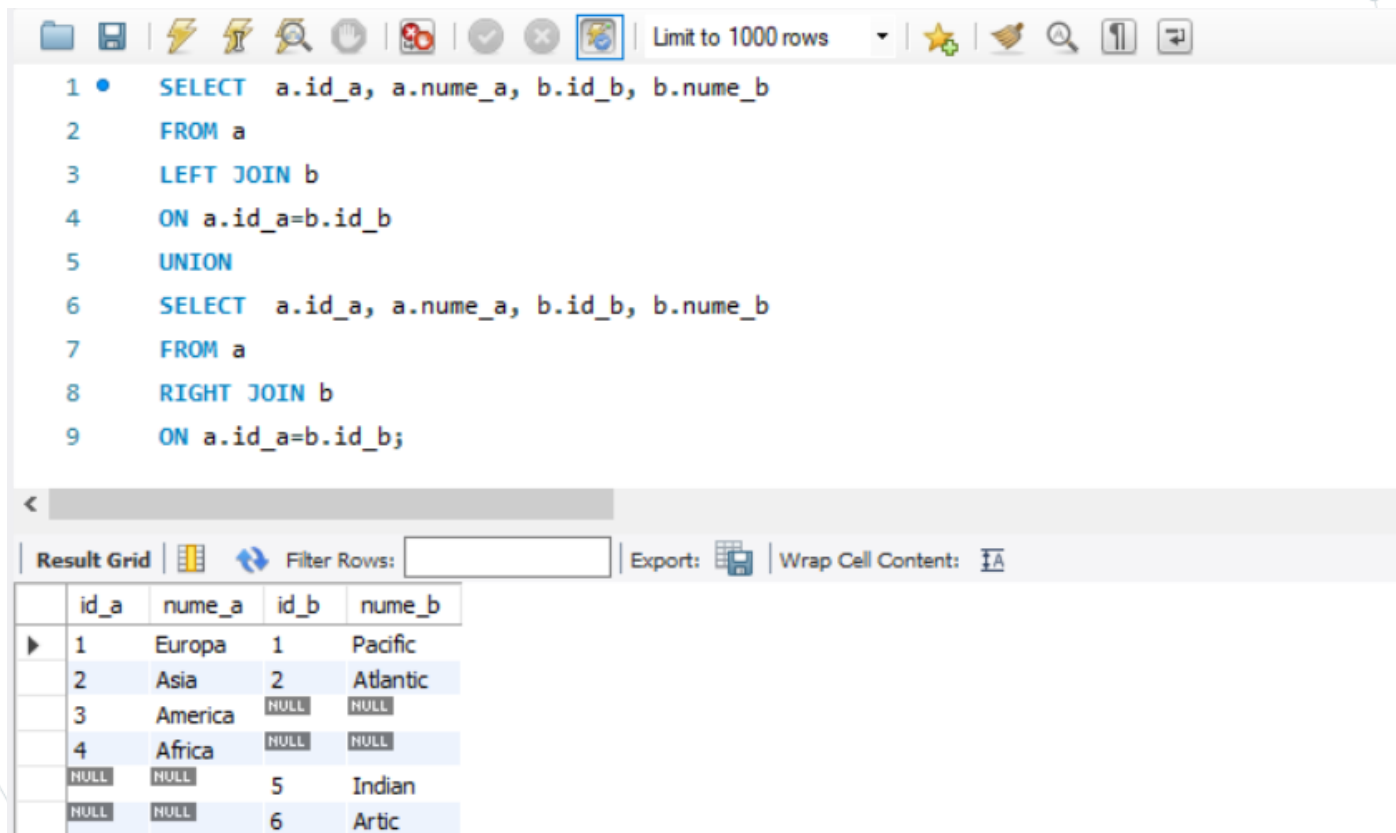
	id_a	nume_a	id_b	nume_b
1	1	Europa	1	Pacific
2	2	Asia	2	Atlantic
	NULL	NULL	5	Indian
	NULL	NULL	6	Artic

Prezentarea asocierii FULL JOIN

- **Prezentarea datelor din 2 tabele asociate după egalitatea valorile a două coloane cu includerea tuturor datelor din ambele tabele**

```
SELECT nume_tabel_stânga.nume_coloane, nume_tabel_dreapta.nume_coloane
FROM nume_tabel_stânga
LEFT JOIN nume_tabel_dreapta
ON nume_tabel_dreapta.nume_coloana_asociere=
nume_tabel_stânga.nume_coloana_asociere
UNION
SELECT nume_tabel_stânga.nume_coloane, nume_tabel_dreapta.nume_coloane
FROM nume_tabel_stânga
RIGHT JOIN nume_tabel_dreapta
ON nume_tabel_dreapta.nume_coloana_asociere=
nume_tabel_stânga.nume_coloana_asociere;
```

Exemplu de asociere FULL JOIN



The screenshot displays a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • SELECT a.id_a, a.numa_a, b.id_b, b.numa_b
2 FROM a
3 LEFT JOIN b
4 ON a.id_a=b.id_b
5 UNION
6 SELECT a.id_a, a.numa_a, b.id_b, b.numa_b
7 FROM a
8 RIGHT JOIN b
9 ON a.id_a=b.id_b;
```

Below the query editor is the "Result Grid" section, which includes a "Filter Rows:" input field, an "Export:" button, and a "Wrap Cell Content:" checkbox. The result grid shows the following data:

	id_a	numa_a	id_b	numa_b
▶ 1	1	Europa	1	Pacific
2	2	Asia	2	Atlantic
3	3	America	NULL	NULL
4	4	Africa	NULL	NULL
5	NULL	NULL	5	Indian
6	NULL	NULL	6	Arctic

Prezentarea tuturor datelor a 2 tabele relaționale

- Prezentarea tuturor datelor a 2 tabele relaționale prin asocierea cheii străine cu cheia primară

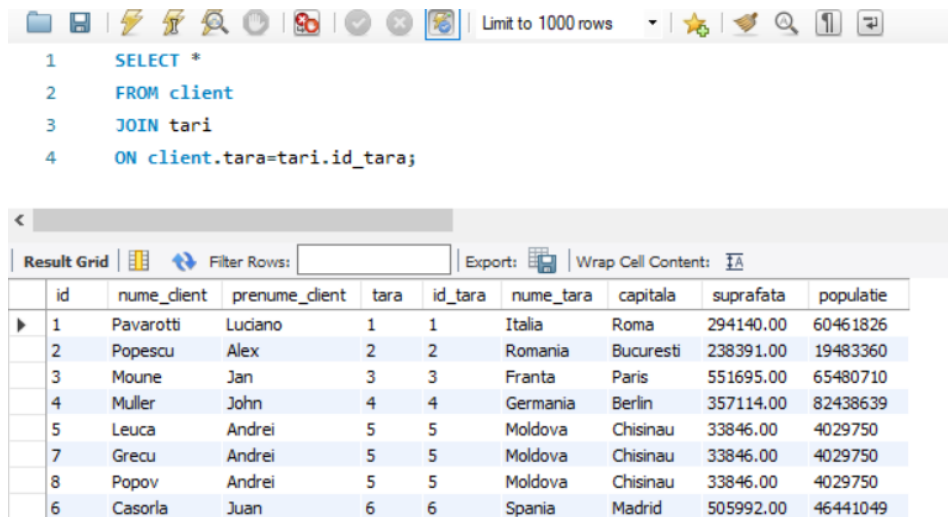
```
SELECT *
```

```
FROM nume_tabel_stânga
```

```
JOIN nume_tabel_dreapta
```

```
ON nume_tabel_dreapta.nume_coloana_cheie_straina=  
nume_tabel_stânga.nume_coloana_cheie_primara;
```

- Exemplu



The screenshot shows a SQL query editor with a toolbar at the top. The query text is as follows:

```
1 SELECT *  
2 FROM client  
3 JOIN tari  
4 ON client.tara=tari.id_tara;
```

Below the query editor is the 'Result Grid' showing the results of the query. The grid has 10 columns: id, nume_client, prenume_client, tara, id_tara, nume_tara, capitala, suprafata, and populatie. There are 6 rows of data.

	id	nume_client	prenume_client	tara	id_tara	nume_tara	capitala	suprafata	populatie
1	1	Pavarotti	Luciano	1	1	Italia	Roma	294140.00	60461826
2	2	Popescu	Alex	2	2	Romania	Bucuresti	238391.00	19483360
3	3	Moune	Jan	3	3	Franta	Paris	551695.00	65480710
4	4	Muller	John	4	4	Germania	Berlin	357114.00	82438639
5	5	Leuca	Andrei	5	5	Moldova	Chisinau	33846.00	4029750
7	7	Grecu	Andrei	5	5	Moldova	Chisinau	33846.00	4029750
8	8	Popov	Andrei	5	5	Moldova	Chisinau	33846.00	4029750
6	6	Casoria	Juan	6	6	Spania	Madrid	505992.00	46441049

Asocierea mai multor tabele

- Pentru asocierea mai multor tabele se utilizează mai multe perechi de cuvinte cheie JOIN și ON iar criteriul de asociere se poate realiza pe coloanele oricărui tabel

```
SELECT nume_tabel_1.nume_col, nume_tabel_2.nume_col, nume_tabel_3.nume_col  
FROM nume_tabel_1  
JOIN nume_tabel_2  
ON nume_tabel_1.nume_col_asociere= nume_tabel_2.nume_col_asociere  
JOIN nume_tabel_3  
ON nume_tabel_1.nume_col_asociere= nume_tabel_3.nume_col_asociere;
```

6. Instrucțiunea UPDATE

Actualizarea datelor conform cheii primare

- Instrucțiunea UPDATE – permite actualizarea datelor in baza de date
- SET – permite specificarea valorii actualizate
- Modificarea datelor unei coloane precizându-se valoarea cheii primare a liniei

```
UPDATE nume_tabel SET nume_coloana=valoarea_nouă  
WHERE nume_coloana_cheie_primara=valoarea_cheie_primara;
```

- Exemplu



Actualizarea datelor conform valorilor coloanei

- **SQL_SAFE_UPDATES** – variabilă ce interzice actualizarea sau ștergerea datelor dacă nu se specifică cheia primară (implicit SQL_SAFE_UPDATES=1)
- Modificarea datelor pe mai multe coloane precizându-se valorile datelor unor coloane pe linia respectiva (necesita modificarea variabilei SQL_SAFE_UPDATES)

```
SET SQL_SAFE_UPDATES = 0;
```

```
UPDATE nume_tabel SET nume_col_1=valoare_col_1_nou, nume_col_2=valoare_col_2_nou  
WHERE nume_col_1=valoare_col_1_vechi AND nume_col_2=valoare_col_2_vechi;
```

```
SET SQL_SAFE_UPDATES = 1;
```

- **Exemplu**

```
1 • SET SQL_SAFE_UPDATES = 0;  
2 • UPDATE client SET nume_client='Lupu', prenume_client='Vasile'  
3   WHERE nume_client='Popov' AND prenume_client='Andrei';  
4 • SET SQL_SAFE_UPDATES = 1;
```

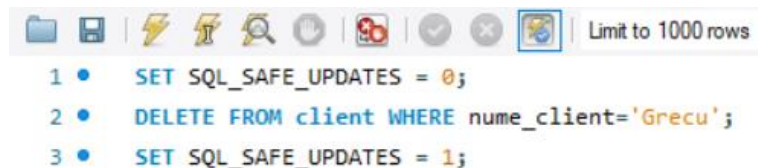
7. Instrucțiunea DELETE

Ștergerea datelor dintr-un tabel

- Instrucțiunea DELETE – permite ștergerea datelor in baza de date
- Ștergerea înscrieri precizându-se valoarea unei coloane a acesteia

```
DELETE FROM nume_tabel WHERE nume_col=valoarea_col;
```

- Exemplu



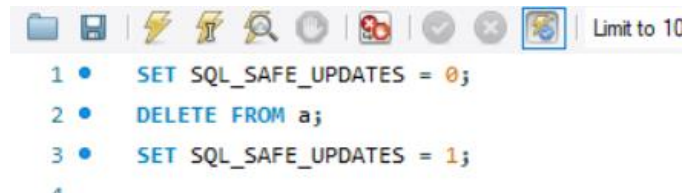
A screenshot of a SQL query editor window. The toolbar at the top includes icons for file operations, execution, and search, along with a 'Limit to 1000 rows' option. The query text is as follows:

```
1 • SET SQL_SAFE_UPDATES = 0;  
2 • DELETE FROM client WHERE nume_client='Greco';  
3 • SET SQL_SAFE_UPDATES = 1;
```

- Ștergerea tuturor datelor unui tabel

```
DELETE FROM nume_tabel
```

- Exemplu



A screenshot of a SQL query editor window. The toolbar at the top includes icons for file operations, execution, and search, along with a 'Limit to 10' option. The query text is as follows:

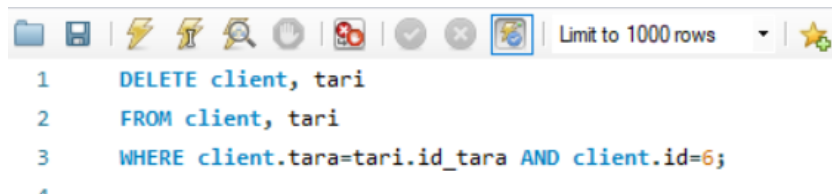
```
1 • SET SQL_SAFE_UPDATES = 0;  
2 • DELETE FROM a;  
3 • SET SQL_SAFE_UPDATES = 1;
```

Ștergerea datelor în tabele relaționale

- Ștergerea unei înregistrări dintr-un tabel cu cheie străină prin ștergerea și a datelor din tabelul cu cheie primara

```
DELETE nume_tabel_1, nume_tabel_2
FROM nume_tabel_1, nume_tabel_2
WHERE nume_tabel_1.nume_col_cheie_strain = nume_tabel_2.nume_col_cheie_primar
AND nume_tabel_1.nume_col=valoarea_col
```

- Exemplu

A screenshot of a SQL query editor window. The window has a toolbar at the top with icons for file operations, execution, and search. Below the toolbar, the SQL query is displayed in a monospaced font. The query is a DELETE statement that removes records from two tables, 'client' and 'tari', based on a foreign key relationship. The query is as follows:

```
1 DELETE client, tari
2 FROM client, tari
3 WHERE client.tara=tari.id_tara AND client.id=6;
```