

Baze de date

Limbajul SQL

Teams: FI-AIA-2-Baze de date-2022-2023



THE **INFORMATION** COMPANY

Curs 4

Limbajul SQL

Limbajul SQL

Interogări SELECT pe o singură tabelă (partea II)

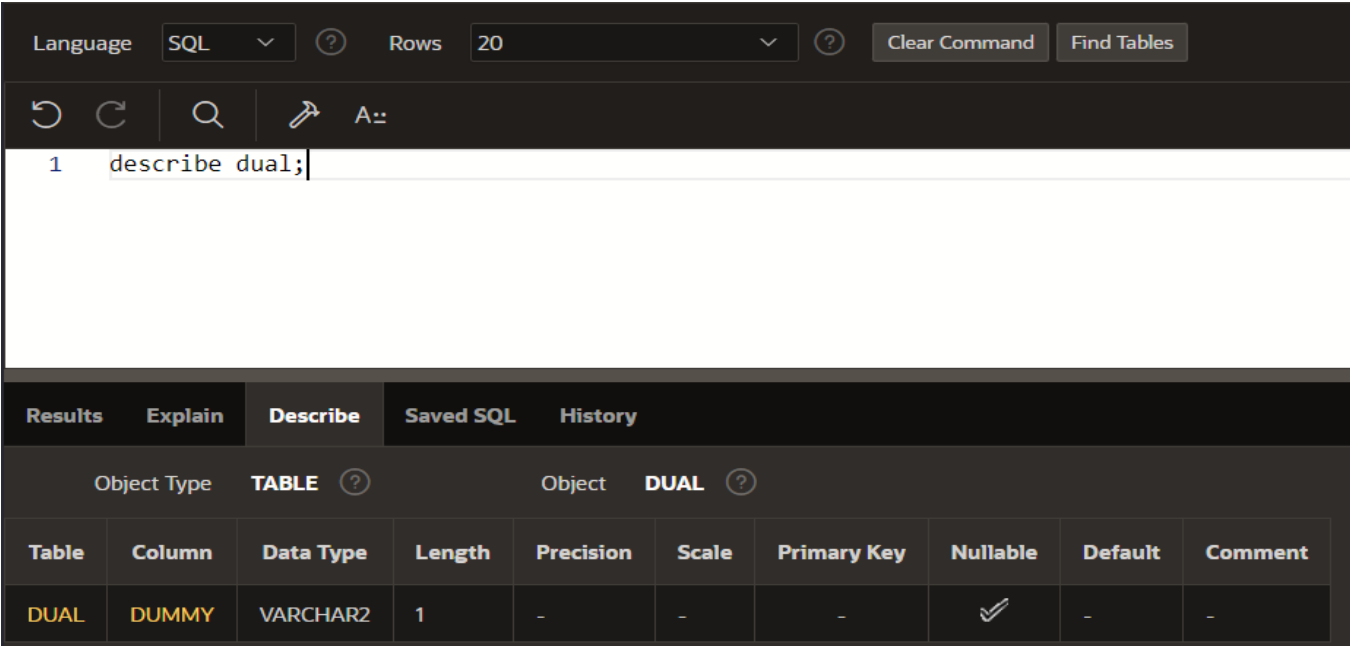
4.1. Funcții

4.2. Funcții referitoare la o singură înregistrare

Tabela DUAL

Tabela generica **DUAL** se foloseste pentru a *testa functii* si pentru a *evalua diferite expresii* care nu necesita preluarea datelor dintr-o tabela.

Această tabela este una specială, care conține o singură coloană numită "**DUMMY**" și o singură linie.



The screenshot shows a database client interface. At the top, there's a toolbar with 'Language' set to 'SQL', 'Rows' set to '20', and buttons for 'Clear Command' and 'Find Tables'. Below the toolbar is a command input area with the text '1 describe dual;'. The main area displays the results of the command in a table format. The table has columns for 'Table', 'Column', 'Data Type', 'Length', 'Precision', 'Scale', 'Primary Key', 'Nullable', 'Default', and 'Comment'. The single row shows the 'DUAL' table with a 'DUMMY' column of type 'VARCHAR2' and length '1'. The 'Nullable' column has a checkmark icon.

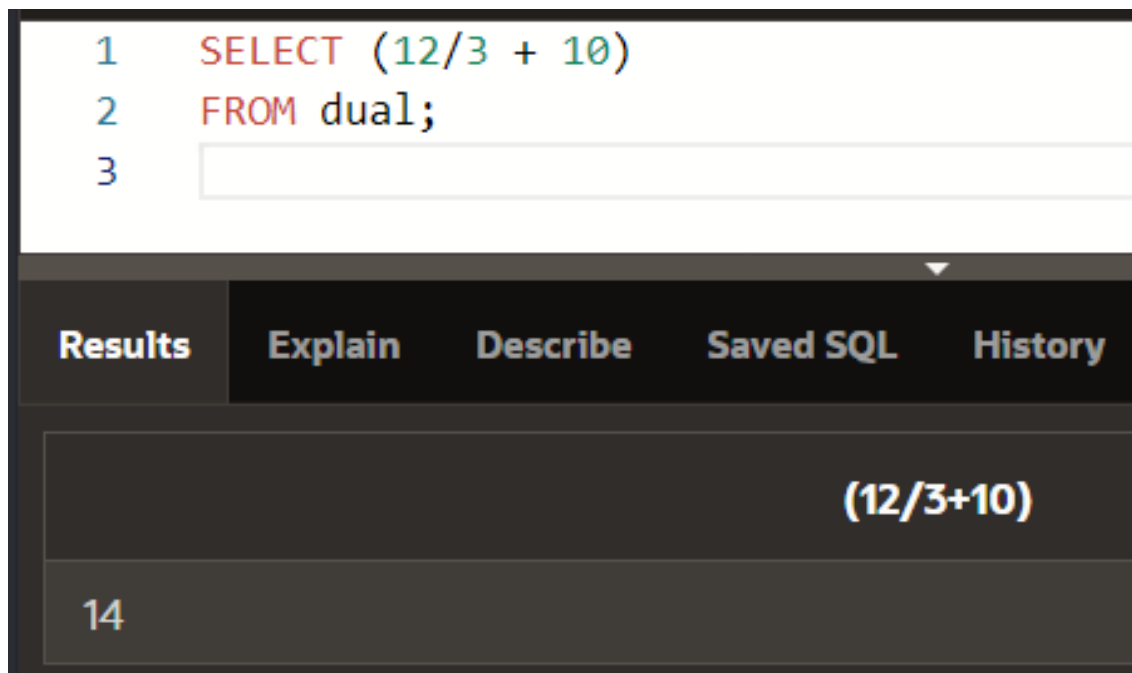
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DUAL	DUMMY	VARCHAR2	1	-	-	-	✓	-	-

Tabela DUAL

Putem folosi tabela **DUAL** si atunci cand vrem sa realizăm diverse calcule.

Exemplu:

```
SELECT (12/3 + 10)  
FROM dual;
```



The screenshot shows a SQL query execution interface. The query is entered in a text area and is as follows:

```
1  SELECT (12/3 + 10)  
2  FROM dual;  
3
```

Below the query area, there are tabs for "Results", "Explain", "Describe", "Saved SQL", and "History". The "Results" tab is selected, and the results are displayed in a table with two rows:

(12/3+10)
14

Funcții

Funcțiile sunt o caracteristică importantă a
SQL si sunt utilizate pentru:

- ✓ a realiza calcule asupra datelor
- ✓ a modifica date
- ✓ a manipula grupuri de înregistrări(linii)
- ✓ a schimba formatul datelor
- ✓ sau pentru a converti diferite tipuri de date

Funcții referitoare la o singură înregistrare

În documentația **ORACLE** puteți găsi foarte multe funcții care pot fi utilizate în expresii.

Lista completă a acestor funcții este la:

<https://docs.oracle.com/cloud/help/o/analytics-cloud/ACUBI/GUID-4CBCE8D4-CF17-43BD-AAEF-C5D614A8040A.htm#BILUG779>

- Funcțiile de agregare
- Funcții pt. analize
- Funcții dată și oră
- Funcțiile de conversie
- Funcții de afișare
- Funcțiile de evaluare
- Funcțiile matematice
- Rularea funcțiilor de agregare
- Funcții spațiale
- Funcții pt. șiruri
- Funcții de sistem
- Funcțiile pt. serii temporale

Funcții

Funcțiile se pot clasifica în două categorii:

1. Funcții referitoare la o singură înregistrare
(single-row functions)
2. Funcții referitoare la mai multe înregistrări
(multiple-row functions)

Funcții

1. Funcții referitoare la o singură înregistrare (single-row functions):

1. Funcții pentru șiruri de caractere
2. Funcții de tip numeric
3. Funcții de tip dată calendaristică și oră
4. Funcții de conversie dintr-un tip în altul
5. Funcții generale
6. Funcții condiționale

Funcții

2. Funcții referitoare la mai multe înregistrări (multiple-row functions):

- Funcții totalizatoare sau funcții de grup

Funcții

Diferența dintre cele două tipuri de funcții este numărul de înregistrări pe care acționează:

- *Funcțiile referitoare la o singură înregistrare returnează un singur rezultat pentru fiecare rând al tablei,*
- *pe când funcțiile referitoare la mai multe înregistrări returnează un singur rezultat pentru fiecare grup de înregistrări din tabela.*

Funcții

O observație importantă este faptul că dacă se apelează o funcție **SQL** ce are un argument (parametru) egal cu valoarea **Null**, atunci în mod automat rezultatul va avea valoarea **Null**.

Singurele funcții care nu respectă această regulă sunt:

- **CONCAT**
- **DECODE**
- **DUMP**
- **NVL**
- **REPLACE**

Limbajul SQL

Interogări SELECT pe o singură tabelă (partea II)

4.1. Funcții

4.2. Funcții referitoare la o singură înregistrare

Funcții referitoare la o singură înregistrare

Funcțiile referitoare la o singură înregistrare pot fi folosite în:

- a) clauza **SELECT** - pentru a modifica modul de afișare a datelor, pentru a realiza diferite calcule, etc.
- b) clauza **WHERE** - pentru a scrie condiția pe baza careia se afișează înregistrările(liniile)
- c) clauza **ORDER BY** - pentru a afișa datele pe baza unor criterii de sortare

Limbajul SQL

Interogări SELECT pe o singură tabelă (partea II)

4.2. Funcții referitoare la o singură înregistrare

4.2.1. Funcții pentru șiruri de caractere

4.2.2. Funcții de tip numeric

4.2.3. Funcții de tip dată calendaristică și oră

4.2.4. Funcții de conversie dintr-un tip în altul

4.2.5. Funcții generale

4.2.6. Funcții condiționale

Funcții referitoare la o singură înregistrare

4.2.1. Funcții pentru șiruri de caractere

Aceste funcții au ca argumente date de tip caracter și returnează date de tip **VARCHAR2**, **CHAR** sau **NUMBER**.

4.2.1. Funcții pentru șiruri de caractere

Cele mai importante funcții caracter sunt:

Funcție	Descriere
LOWER(column expression)	converteste alfa caracterele din caractere mari in caractere mici
UPPER(column expression)	converteste alfa caracterele din caractere mici in caractere mari
INITCAP(column expression)	converteste prima litera a fiecarui cuvant in caractere mari si restul cuvantului in caractere mici
CONCAT(column1 expression1, column2 expression2)	functia este echivalentul operatorului de concantenare ()
SUBSTR(column expression, m [, n])	returneaza un sir de <i>n</i> caractere incepand cu caracterul aflat pe pozitia <i>m</i>
LENGTH(column expression)	returneaza numarul de caractere dintr-o expresie
INSTR(column expression, 'string', [m], [n])	returneaza pozitia unui anumit sir, optional se poate incepe cautarea cu pozitia <i>m</i> sau cu a <i>n</i> -a aparitie a sirului. <i>m</i> si <i>n</i> sunt prin definitie 1
REPLACE(text, search_string, replacement_string)	cauta un anumit text intr-un sir de caractere si daca il gaseste il inlocuieste

4.2.1. Funcții pentru șiruri de caractere

Exemplu de utilizare a funcției **LENGTH**:

```
SELECT LENGTH(ename)  
FROM EMP;
```

Column Name	Data Type	Nullable	Default	Primary Key
EMPNO	NUMBER(4,0)	No		1
ENAME	VARCHAR2(50)	Yes		
JOB	VARCHAR2(50)	Yes		
MGR	NUMBER(4,0)	Yes		
HIREDATE	DATE	Yes		
SAL	NUMBER(7,2)	Yes		
COMM	NUMBER(7,2)	Yes		
DEPTNO	NUMBER(2,0)	Yes		

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	5/1/1981	2850		30
7782	CLARK	MANAGER	7839	6/9/1981	2450		10
7566	JONES	MANAGER	7839	4/2/1981	2975		20
7788	SCOTT	ANALYST	7566	12/9/1982	3000		20
7902	FORD	ANALYST	7566	12/3/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	2/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	2/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	9/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	9/8/1981	1500	0	30
7876	ADAMS	CLERK	7788	1/12/1983	1100		20
7900	JAMES	CLERK	7698	12/3/1981	950		30
7934	MILLER	CLERK	7782	1/23/1982	1300		10

4.2.1. Funcții pentru șiruri de caractere

Exemplu de utilizare a funcției **LENGTH** – rezultatul obținut:

```
1  SELECT LENGTH(ename)  
2  FROM EMP;
```

Results	Explain	Describe	Saved SQL	History
LENGTH(ENAME)				
4				
5				
5				
5				
5				
4				
5				
5				
4				
6				
6				
5				
5				
6				

14 rows returned in 0.03 seconds [Download](#)

4.2.1. Funcții pentru șiruri de caractere

Exemplu:

```
SELECT 'Numele functiei pentru ' || UPPER(ename) || 'este  
      ' || LOWER(job) AS "DETALII ANGAJAT"  
FROM EMP;
```

Column Name	Data Type	Nullable	Default	Primary Key
EMPNO	NUMBER(4,0)	No		1
ENAME	VARCHAR2(50)	Yes		
JOB	VARCHAR2(50)	Yes		
MGR	NUMBER(4,0)	Yes		
HIREDATE	DATE	Yes		
SAL	NUMBER(7,2)	Yes		
COMM	NUMBER(7,2)	Yes		
DEPTNO	NUMBER(2,0)	Yes		

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	5/1/1981	2850		30
7782	CLARK	MANAGER	7839	6/9/1981	2450		10
7566	JONES	MANAGER	7839	4/2/1981	2975		20
7788	SCOTT	ANALYST	7566	12/9/1982	3000		20
7902	FORD	ANALYST	7566	12/3/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	2/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	2/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	9/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	9/8/1981	1500	0	30
7876	ADAMS	CLERK	7788	1/12/1983	1100		20
7900	JAMES	CLERK	7698	12/3/1981	950		30
7934	MILLER	CLERK	7782	1/23/1982	1300		10

4.2.1. Funcții pentru șiruri de caractere

Rezultat obtinut:

```
1  SELECT 'Numele functiei pentru '||UPPER(ename)||' este '||LOWER(job) AS "DETALII ANGAJAT"  
2  FROM EMP;
```

Results	Explain	Describe	Saved SQL	History
DETALII ANGAJAT				
Numele functiei pentru KING este president				
Numele functiei pentru BLAKE este manager				
Numele functiei pentru CLARK este manager				
Numele functiei pentru JONES este manager				
Numele functiei pentru SCOTT este analyst				
Numele functiei pentru FORD este analyst				
Numele functiei pentru SMITH este clerk				
Numele functiei pentru ALLEN este salesman				
Numele functiei pentru WARD este salesman				
Numele functiei pentru MARTIN este salesman				
Numele functiei pentru TURNER este salesman				
Numele functiei pentru ADAMS este clerk				
Numele functiei pentru JAMES este clerk				
Numele functiei pentru MILLER este clerk				
14 rows returned in 0.00 seconds Download				

4.2.1. Funcții pentru șiruri de caractere

Exemplu:

```
SELECT empno, UPPER(ename), job, deptno
FROM EMP
WHERE ename = 'MARTIN';
```

Column Name	Data Type	Nullable	Default	Primary Key
EMPNO	NUMBER(4,0)	No		1
ENAME	VARCHAR2(50)	Yes		
JOB	VARCHAR2(50)	Yes		
MGR	NUMBER(4,0)	Yes		
HIREDATE	DATE	Yes		
SAL	NUMBER(7,2)	Yes		
COMM	NUMBER(7,2)	Yes		
DEPTNO	NUMBER(2,0)	Yes		

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	5/1/1981	2850		30
7782	CLARK	MANAGER	7839	6/9/1981	2450		10
7566	JONES	MANAGER	7839	4/2/1981	2975		20
7788	SCOTT	ANALYST	7566	12/9/1982	3000		20
7902	FORD	ANALYST	7566	12/3/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	2/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	2/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	9/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	9/8/1981	1500	0	30
7876	ADAMS	CLERK	7788	1/12/1983	1100		20
7900	JAMES	CLERK	7698	12/3/1981	950		30
7934	MILLER	CLERK	7782	1/23/1982	1300		10

4.2.1. Funcții pentru șiruri de caractere

Rezultat obtinut:

```
1  SELECT empno, UPPER(ename), job, deptno
2  FROM EMP
3  WHERE ename = 'MARTIN';
```

Results Explain Describe Saved SQL History			
EMPNO	UPPER(ENAME)	JOB	DEPTNO
7654	MARTIN	SALESMAN	30
1 rows returned in 0.01 seconds Download			

4.2.1. Funcții pentru șiruri de caractere

Clauza **WHERE** a acestei cereri **SQL** compară numele din tabela Angajați cu 'Smith'.

Pentru comparație numele sunt convertite în litere mici și din această cauză se obține un rezultat.

Exemplu:

```
SELECT empno, UPPER(ename), job, deptno  
FROM EMP  
WHERE INITCAP(ename) = 'Smith';
```


4.2.1. Funcții pentru șiruri de caractere

Rezultatul obtinut:

```
1  SELECT empno, UPPER(ename), job, deptno
2  FROM EMP
3  WHERE INITCAP(ename) = 'Smith';
```

Results Explain Describe Saved SQL History			
EMPNO	UPPER(ENAME)	JOB	DEPTNO
7369	SMITH	CLERK	20
1 rows returned in 0.00 seconds Download			

4.2.1. Funcții pentru șiruri de caractere

Exemplu:

Pentru afișarea numelui cu majuscule de folosește funcția **UPPER**.

```
SELECT empno, CONCAT(ename, job), ename,  
       UPPER(ename)  
FROM EMP;
```

4.2.1. Funcții pentru șiruri de caractere

Rezultatul obtinut:

```
1  SELECT empno, CONCAT(ename, job), ename, UPPER(ename)
2  FROM EMP;
```

Results

Explain

Describe

Saved SQL

History

EMPNO	CONCAT(ENAME, JOB)	ENAME	UPPER(ENAME)
7839	KINGPRESIDENT	KING	KING
7698	BLAKEMANAGER	BLAKE	BLAKE
7782	CLARKMANAGER	CLARK	CLARK
7566	JONESMANAGER	JONES	JONES
7788	SCOTTANALYST	SCOTT	SCOTT
7902	FORDANALYST	FORD	FORD
7369	SMITHCLERK	SMITH	SMITH
7499	ALLENSALESMAN	ALLEN	ALLEN
7521	WARDSALESMAN	WARD	WARD
7654	MARTINSALESMAN	MARTIN	MARTIN
7844	TURNERSALESMAN	TURNER	TURNER
7876	ADAMSCLERK	ADAMS	ADAMS
7900	JAMESCLERK	JAMES	JAMES
7934	MILLERCLERK	MILLER	MILLER

14 rows returned in 0.01 seconds

Download

4.2.1. Funcții pentru șiruri de caractere

- Spre deosebire de alte funcții, funcțiile caracter pot fi imbricate până la orice adâncime.
- Dacă funcțiile sunt imbricate, atunci ele sunt **evaluate din interior spre exterior**.
- Pentru a determina, de exemplu, *de câte ori apare caracterul 'A'* în câmpul *ename* vom folosi interogarea:

4.2.1. Funcții pentru șiruri de caractere

SELECT ename, **LENGTH** (ename) - **LENGTH** (**TRANSLATE**(ename, 'DA', 'D'))
FROM EMP;

Column Name	Data Type	Nullable	Default	Primary Key
EMPNO	NUMBER(4,0)	No		1
ENAME	VARCHAR2(50)	Yes		
JOB	VARCHAR2(50)	Yes		
MGR	NUMBER(4,0)	Yes		
HIREDATE	DATE	Yes		
SAL	NUMBER(7,2)	Yes		
COMM	NUMBER(7,2)	Yes		
DEPTNO	NUMBER(2,0)	Yes		

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	5/1/1981	2850		30
7782	CLARK	MANAGER	7839	6/9/1981	2450		10
7566	JONES	MANAGER	7839	4/2/1981	2975		20
7788	SCOTT	ANALYST	7566	12/9/1982	3000		20
7902	FORD	ANALYST	7566	12/3/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	2/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	2/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	9/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	9/8/1981	1500	0	30
7876	ADAMS	CLERK	7788	1/12/1983	1100		20
7900	JAMES	CLERK	7698	12/3/1981	950		30
7934	MILLER	CLERK	7782	1/23/1982	1300		10

4.2.1. Funcții pentru șiruri de caractere

Rezultatul obtinut:

```
1  SELECT ename, LENGTH (ename) - LENGTH (TRANSLATE(ename, 'DA', 'D'))
2  FROM EMP;
```

Results		Explain	Describe	Saved SQL	History
ENAME		LENGTH(ENAME)-LENGTH(TRANSLATE(ENAME,'DA','D'))			
KING		0			
BLAKE		1			
CLARK		1			
JONES		0			
SCOTT		0			
FORD		0			
SMITH		0			
ALLEN		1			
WARD		1			
MARTIN		1			
TURNER		0			
ADAMS		2			
JAMES		1			
MILLER		0			

14 rows returned in 0.00 seconds [Download](#)

4.2.1. Funcții pentru șiruri de caractere

Explicatii:

În exemplul anterior, funcția **TRANSLATE** (nume, 'DA', 'D') va căuta în coloana “nume” primul caracter (caracterul 'D') din cel de-al doilea argument al funcției (șirul de caractere 'DA') și îl va înlocui cu primul caracter (adică tot cu caracterul 'D') din cel de-al treilea argument al funcției (șirul de caractere 'D'), apoi va căuta cel de-al doilea caracter, adică caracterul 'A', și îl va șterge din câmpul nume deoarece acesta nu are caracter corespondent în cel de-al treilea argument al funcției.

Am folosit acest artificiu deoarece șirul de caractere vid este echivalent cu valoarea **Null**, deci funcția **TRANSLATE** (nume, 'A', ' ') ar fi înlocuit toate valorile câmpului “nume” cu valoarea **Null**.

Limbajul SQL

Interogări SELECT pe o singură tabelă (partea II)

4.2. Funcții referitoare la o singură înregistrare

4.2.1. Funcții pentru șiruri de caractere

4.2.2. Funcții de tip numeric

4.2.3. Funcții de tip dată calendaristică și oră

4.2.4. Funcții de conversie dintr-un tip în altul

4.2.5. Funcții generale

4.2.6. Funcții condiționale

4.2.2. Funcții de tip numeric

Aceste funcții au ca argumente date de tip **NUMBER** și returnează date de tip **numeric**.

4.2.2. Funcții de tip numeric

Cele mai importante funcții pentru valori numerice sunt:

ABS(n)

SIN(n), COS(n), TAN(n)

ACOS(n), ASIN(n), ATAN(n)

POWER(m, n)

SQRT(x)

REMAINDER(x, y)

MOD(a, b)

SIGN(x)

CEIL(x)

FLOOR(x)

ROUND(a, b)

TRUNC(a, b)

4.2.2. Funcții de tip numeric

ABS(n) returnează valoarea absolută a argumentului

Exemple:

```
1 select abs(-12.34), abs(7) from dual
2
3 |
```

Results

Explain

Describe

Saved SQL

History

ABS(-12.34)

ABS(7)

12.34

7

4.2.2. Funcții de tip numeric

SIN(n), COS(n), TAN(n), ACOS(n), ASIN(n), ATAN(n) - sunt funcțiile trigonometrice cu aceeași semnificație ca și la matematică.

Argumentul acestor funcții trebuie precizat în radiani

Exemple:

`select sin(3.1415/4), cos(3.1415/4), tan(3.1415/4) from dual`

<pre>1 select sin(3.1415/4), cos(3.1415/4), tan(3.1415/4) from dual 2 3</pre>		
Results Explain Describe Saved SQL History		
SIN(3.1415/4)	COS(3.1415/4)	TAN(3.1415/4)
.707090402001441379926423870107673024501	.707123159992260488382758141389523768594	.9999536742781562026694237274504983726

`select asin(3.1415/4), acos(3.1415/4), atan(3.1415/4) from dual`

<pre>1 select asin(3.1415/4), acos(3.1415/4), atan(3.1415/4) from dual 2 3</pre>		
Results Explain Describe Saved SQL History		
ASIN(3.1415/4)	ACOS(3.1415/4)	ATAN(3.1415/4)
.903301690429885817654436129706007384622	.667494636365010801576885561933744057478	.665759423619510103191969431353239944313

4.2.2. Funcții de tip numeric

POWER(m, n) - calculează valoarea m^n .

Exemple:

```
select power(2,5), power(2,0.5), power(2,-1),  
       power(2,-0.75) from dual
```

```
1 select power(2,5), power(2,0.5), power(2,-1), power(2,-0.75) from dual  
2  
3  
4
```

Results

Explain

Describe

Saved SQL

History

POWER(2,5)

POWER(2,0.5)

POWER(2,-1)

POWER(2,-0.75)

32

1.41421356237309504880168872420969807855

.5

.594603557501360533358749985280237957651

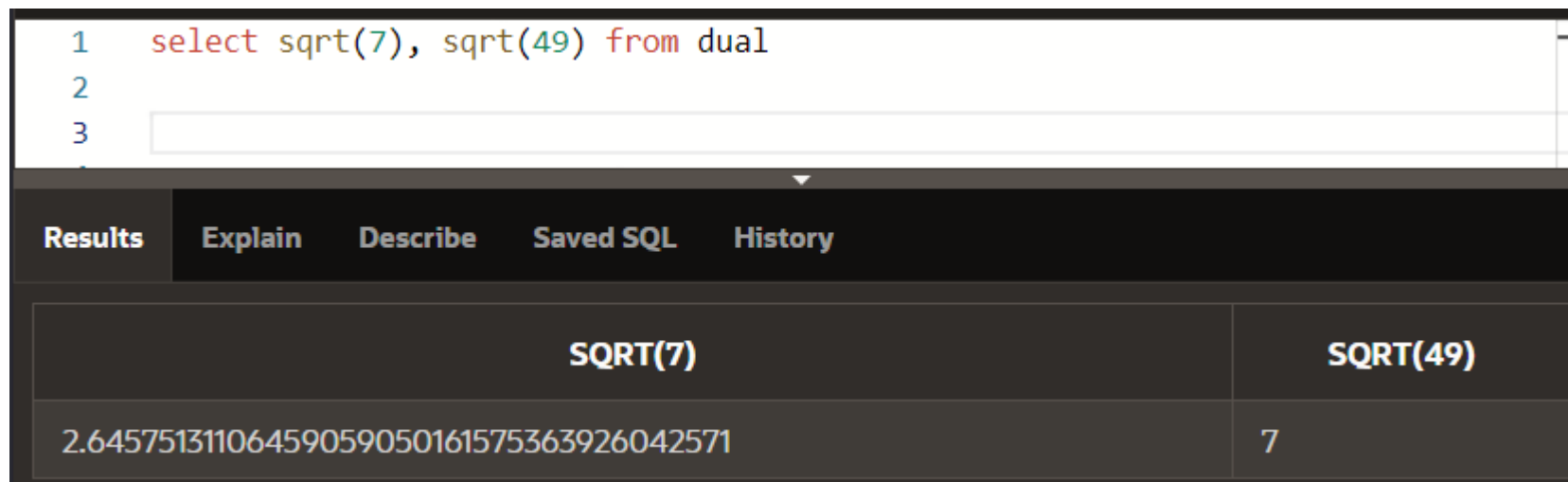
4.2.2. Funcții de tip numeric

SQRT(x) - calculează rădăcina pătrată a argumentului.

Apelul SQRT(x) returnează aceeași valoare ca și POWER(x, 0.5).

Exemple:

`select sqrt(7), sqrt(49) from dual`



The screenshot shows a SQL query execution interface. At the top, the query `select sqrt(7), sqrt(49) from dual` is entered. Below the query, there are tabs for **Results**, **Explain**, **Describe**, **Saved SQL**, and **History**. The **Results** tab is selected, displaying a table with two columns: **SQRT(7)** and **SQRT(49)**. The first row of data shows the values `2.64575131106459059050161575363926042571` and `7`.

SQRT(7)	SQRT(49)
2.64575131106459059050161575363926042571	7

4.2.2. Funcții de tip numeric

REMAINDER(x,y) - în cazul în care ambii parametri x și y sunt numere întregi, funcția calculează restul împărțirii lui x la y. Dacă cel puțin unul dintre parametri este număr real, funcția determină mai întâi acel multiplu a lui y care este cel mai apropiat de x, și returnează apoi diferența dintre x și acel multiplu

Exemple:

**select remainder(10,3), remainder(5,3), remainder(10,3.5),
remainder(-10,3.5) from dual**

1

select remainder(10,3), remainder(5,3), remainder(10,3.5), remainder(-10,3.5) from dual

2

3

Results

Explain

Describe

Saved SQL

History

REMAINDER(10,3)	REMAINDER(5,3)	REMAINDER(10,3.5)	REMAINDER(-10,3.5)
1	-1	-.5	.5

4.2.2. Funcții de tip numeric

MOD(a, b) - dacă cei doi parametri sunt numere întregi, atunci funcția returnează același rezultat ca și funcția **REMAINDER**, adică restul împărțirii lui a la b.

Teorema împărțirii cu rest este extinsă de această funcție și pentru numerele reale.

Adică se ține cont de relația **$a = b * \text{cât} + \text{rest}$** unde restul în modul, trebuie să fie strict mai mic decât b

Exemple:

```
1 select mod(10,3), mod(5,3), mod(10,3.5), mod(-10,3), mod(-10,-3.5), mod(10,-3.5) from dual
2
3
```

Results Explain Describe Saved SQL History

MOD(10,3)	MOD(5,3)	MOD(10,3.5)	MOD(-10,3)	MOD(-10,-3.5)	MOD(10,-3.5)
1	2	3	-1	-3	3

4.2.2. Funcții de tip numeric

MOD(a, b) (continuare)

Exemple:

**select mod(10,3), mod(5,3), mod(10,3.5), mod(-10,3),
mod(-10,-3.5), mod(10,-3.5) from dual**

```
1 select mod(10,3), mod(5,3), mod(10,3.5), mod(-10,3), mod(-10,-3.5), mod(10,-3.5) from dual
2
3
```

Results Explain Describe Saved SQL History

MOD(10,3)	MOD(5,3)	MOD(10,3.5)	MOD(-10,3)	MOD(-10,-3.5)	MOD(10,-3.5)
1	2	3	-1	-3	3

SIGN(x) - returnează semnul lui x, adică 1 dacă x este număr pozitiv, respectiv -1 dacă x este număr negativ.

CEIL(x) - returnează cel mai mic număr întreg care este mai mare sau egal decât parametrul transmis.

FLOOR(x) - returnează cel mai mare număr întreg care este mai mic sau egal decât parametrul transmis.

Exemple: **`select ceil(6), ceil(-6), ceil(-6.7), ceil(6.7) from dual`**



The screenshot shows a SQL query editor with the query: `select ceil(6), ceil(-6), ceil(-6.7), ceil(6.7) from dual`. Below the editor, the 'Results' tab is active, displaying a table with four columns: **CEIL(6)**, **CEIL(-6)**, **CEIL(-6.7)**, and **CEIL(6.7)**. The results are 6, -6, -6, and 7 respectively.

CEIL(6)	CEIL(-6)	CEIL(-6.7)	CEIL(6.7)
6	-6	-6	7

`select floor(2), floor(-2), floor(-8.9), floor(8.9) from dual`



The screenshot shows a SQL query editor with the query: `select floor(2), floor(-2), floor(-8.9), floor(8.9) from dual`. Below the editor, the 'Results' tab is active, displaying a table with four columns: **FLOOR(2)**, **FLOOR(-2)**, **FLOOR(-8.9)**, and **FLOOR(8.9)**. The results are 2, -2, -9, and 8 respectively.

FLOOR(2)	FLOOR(-2)	FLOOR(-8.9)	FLOOR(8.9)
2	-2	-9	8

4.2.2. Funcții de tip numeric

ROUND(a, b) - rotunjește valoarea lui a la un număr de cifre precizat prin parametrul b.

Dacă al doilea parametru este un număr pozitiv, atunci se vor păstra din a primele b zecimale, ultima dintre aceste cifre fiind rotunjită, în funcție de următoarea zecimală.

Al doilea argument poate fi o valoare negativă, rotunjirea făcându-se la stânga punctului zecimal.

Cifra a $|b|+1$ din fața punctului zecimal (numărând de la punctul zecimal spre stânga începând cu 1) va fi rotunjită în funcție cifra aflată imediat la dreapta ei.

Primele $|b|$ cifre din stânga punctului zecimal vor deveni 0. Cel de al doilea argument este opțional, în cazul în care nu se precizează, este considerată implicit valoarea 0.

4.2.2. Funcții de tip numeric

Exemple:

```
select round(789.123,2), round(789.126,2), round(789.126,-1),  
       round(789.126,-2), round(789.126,-3), round(789.126,-4),  
       round(789.126,0), round(789.826,0), round(789.826)  
from dual
```

```
1 select round(789.123,2), round(789.126,2), round(789.126,-1), round(789.126,-2), round(789.126,-3), round(789.126,-4), round(789.126,0), round(789.826,0), round(789.826)
2 from dual
3
```

Results Explain Describe Saved SQL History

ROUND(789.123,2)	ROUND(789.126,2)	ROUND(789.126,-1)	ROUND(789.126,-2)	ROUND(789.126,-3)	ROUND(789.126,-4)	ROUND(789.126,0)	ROUND(789.826,0)	ROUND(789.826)
789.12	789.13	790	800	1000	0	789	790	790

4.2.2. Funcții de tip numeric

TRUNC(a, b) - este asemănătoare cu funcția **ROUND**, fără a rotunji ultima cifră.

Exemple:

```
select trunc(789.123,2), trunc(789.126,2), trunc(789.126,-1),  
       trunc(789.126,-2), trunc(789.126,-3), trunc(789.126,-4),  
       trunc(789.126,0), trunc(789.826,0), trunc(789.826)  
from dual
```

```
1 select trunc(789.123,2), trunc(789.126,2), trunc(789.126,-1), trunc(789.126,-2), trunc(789.126,-3), trunc(789.126,-4), trunc(789.126,0), trunc(789.826,0), trunc(789.826)
2 from dual
3 |
```

Results Explain Describe Saved SQL History

TRUNC(789.123,2)	TRUNC(789.126,2)	TRUNC(789.126,-1)	TRUNC(789.126,-2)	TRUNC(789.126,-3)	TRUNC(789.126,-4)	TRUNC(789.126,0)	TRUNC(789.826,0)	TRUNC(789.826)
789.12	789.12	780	700	0	0	789	789	789

Limbajul SQL

Interogări SELECT pe o singură tabelă (partea II)

4.2. Funcții referitoare la o singură înregistrare

4.2.1. Funcții pentru șiruri de caractere

4.2.2. Funcții de tip numeric

4.2.3. Funcții de tip dată calendaristică și oră

4.2.4. Funcții de conversie dintr-un tip în altul

4.2.5. Funcții generale

4.2.6. Funcții condiționale

4.2.3. Funcții de tip dată calendaristică și oră

Aceste funcții au ca argumente date de tip **DATE** și returnează date de tip DATE.

4.2.3. Funcții de tip dată calendaristică și oră

Cele mai importante funcții de tip data calendaristica si timp sunt:

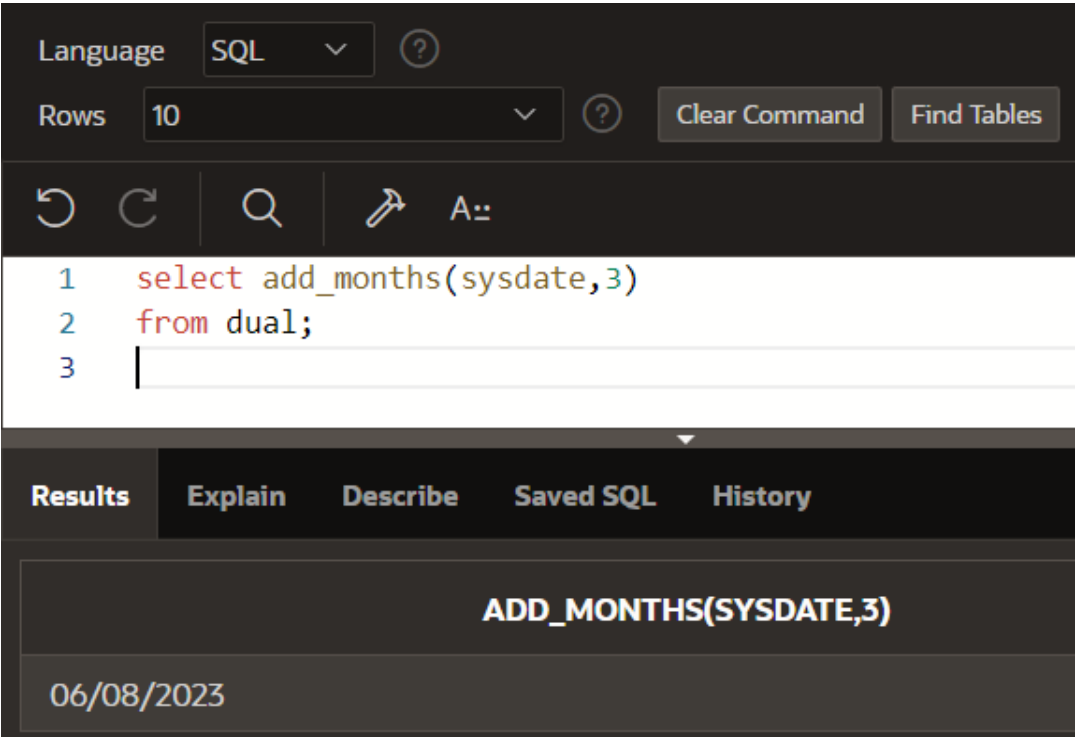
Funcție	Descriere
ADD_MONTHS(column expression, n)	Adauga un numar n de luni unei date calendaristice
LAST_DAY(column expression)	Determina care este ultima zi dintr-o luna
MONTHS_BETWEEN(data_inceput, data_sfarsit)	Determina cate luni sunt intre doua date calendaristice
NEXT_DAY(column expression, char)	Returneaza ziua urmatoare datei transmise ca argument pe baza char
ROUND(column expression)	Rotunjeste data calendaristica
TRUNC(column expression)	Trunchiaza data calendaristica
SYSDATE	Returneaza data calendaristica curenta

<https://docs.oracle.com/cloud/help/ro/analytics-cloud/ACUBI/GUID-4CBCE8D4-CF17-43BD-AAEF-C5D614A8040A.htm#GUID-1A697795-7D1E-4296-961A-1002FDBD4F47>

4.2.3. Funcții de tip dată calendaristică și oră

Funcția **ADD_MONTHS** - exemplu:

```
SELECT ADD_MONTHS(SYSDATE,3)  
FROM DUAL;
```

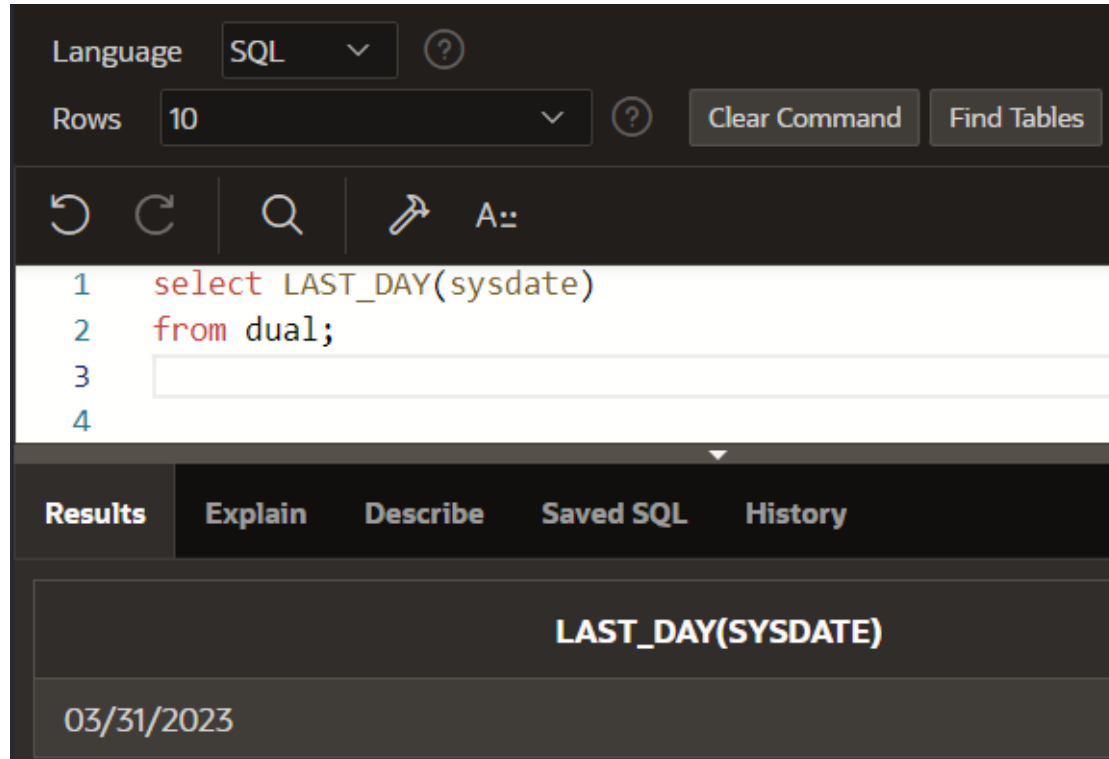


The screenshot shows a SQL IDE interface. At the top, there's a 'Language' dropdown set to 'SQL' and a 'Rows' dropdown set to '10'. Below these are buttons for 'Clear Command' and 'Find Tables'. The main editor area contains the SQL query: `1 select add_months(sysdate,3)`, `2 from dual;`, and `3` followed by a cursor. Below the editor, there's a tabbed interface with 'Results' selected. The results pane shows the output of the query: `ADD_MONTHS(SYSDATE,3)` and the date `06/08/2023`.

4.2.3. Funcții de tip dată calendaristică și oră

Funcția **LAST_DAY** - exemplu:

```
SELECT LAST_DAY(SYSDATE)  
FROM DUAL;
```



The screenshot shows a SQL IDE interface. At the top, there's a 'Language' dropdown set to 'SQL' and a 'Rows' dropdown set to '10'. Below these are buttons for 'Clear Command' and 'Find Tables'. The main editor area contains the SQL query: `1 select LAST_DAY(sysdate)`, `2 from dual;`. Below the editor, there's a tabbed interface with 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, showing a single row with the column header **LAST_DAY(SYSDATE)** and the value `03/31/2023`.

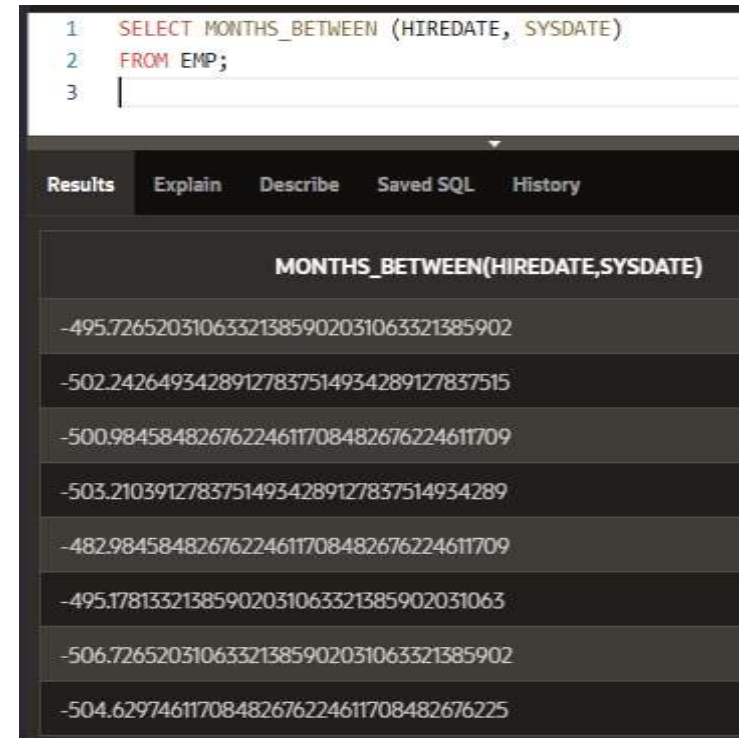
LAST_DAY(SYSDATE)
03/31/2023

4.2.3. Funcții de tip dată calendaristică și oră

Funcția **MONTHS_BETWEEN** - exemplu:

```
SELECT MONTHS_BETWEEN (HIREDATE,  
SYSDATE)  
FROM EMP;
```

Rezultatul va fi un număr negativ,
deoarece primul parametru este o
data calendaristică mai mică decât al
doilea parametru.



The screenshot shows a SQL query execution interface. The query is: `SELECT MONTHS_BETWEEN (HIREDATE, SYSDATE) FROM EMP;`. The results are displayed in a table with a single column titled `MONTHS_BETWEEN(HIREDATE,SYSDATE)`. The results are negative values, indicating that the hire date is earlier than the current date (SYSDATE).

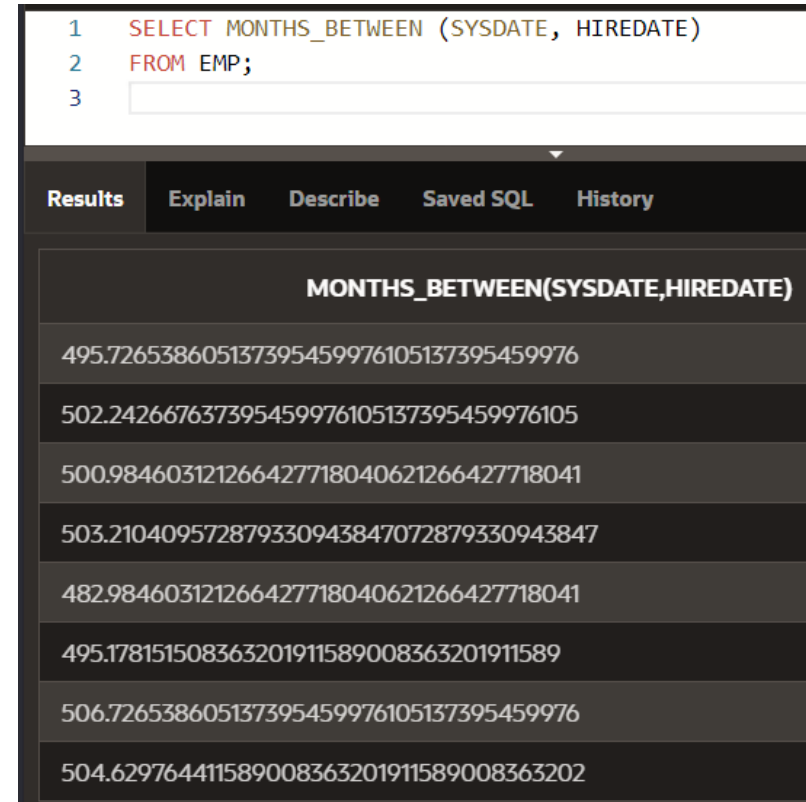
MONTHS_BETWEEN(HIREDATE,SYSDATE)
-495.72652031063321385902031063321385902
-502.24264934289127837514934289127837515
-500.98458482676224611708482676224611709
-503.21039127837514934289127837514934289
-482.98458482676224611708482676224611709
-495.17813321385902031063321385902031063
-506.72652031063321385902031063321385902
-504.62974611708482676224611708482676225

4.2.3. Funcții de tip dată calendaristică și oră

Funcția **MONTHS_BETWEEN** -
exemplu:

```
SELECT MONTHS_BETWEEN  
      (SYSDATE, HIREDATE)  
FROM EMP;
```

Pentru a afișa o valoare pozitivă, se
pot inversa cei doi parametri:



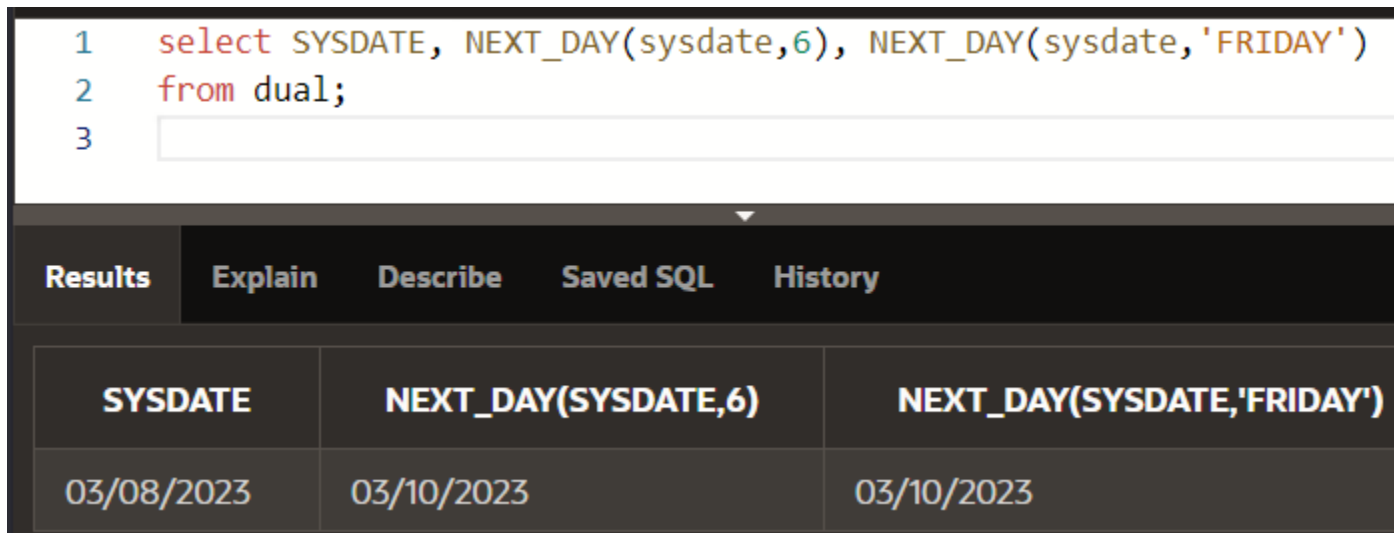
The screenshot shows a SQL query execution interface. The query is: `SELECT MONTHS_BETWEEN (SYSDATE, HIREDATE) FROM EMP;`. The results are displayed in a table with a single column header `MONTHS_BETWEEN(SYSDATE,HIREDATE)`. The results are numerical values representing the number of months between the current date and the hire date for each employee.

MONTHS_BETWEEN(SYSDATE,HIREDATE)
495.726538605137395459976105137395459976
502.242667637395459976105137395459976105
500.984603121266427718040621266427718041
503.210409572879330943847072879330943847
482.984603121266427718040621266427718041
495.178151508363201911589008363201911589
506.726538605137395459976105137395459976
504.629764411589008363201911589008363202

4.2.3. Funcții de tip dată calendaristică și oră

Funcția **NEXT_DAY** - exemplu:

```
SELECT SYSDATE, NEXT_DAY(SYSDATE,6),  
       NEXT_DAY(SYSDATE,'FRIDAY')  
FROM DUAL;
```



The screenshot shows a SQL query execution interface. The query is entered in a text area and is as follows:

```
1 select SYSDATE, NEXT_DAY(sysdate,6), NEXT_DAY(sysdate,'FRIDAY')  
2 from dual;  
3
```

Below the query area, there is a tabbed interface with the following tabs: Results, Explain, Describe, Saved SQL, and History. The 'Results' tab is selected, and it displays the following table:

SYSDATE	NEXT_DAY(SYSDATE,6)	NEXT_DAY(SYSDATE,'FRIDAY')
03/08/2023	03/10/2023	03/10/2023

Limbajul SQL

Interogări SELECT pe o singură tabelă (partea II)

4.2. Funcții referitoare la o singură înregistrare

4.2.1. Funcții pentru șiruri de caractere

4.2.2. Funcții de tip numeric

4.2.3. Funcții de tip dată calendaristică și oră

4.2.4. Funcții de conversie dintr-un tip în altul

4.2.5. Funcții generale

4.2.6. Funcții condiționale

4.2.4. Funcții de conversie dintr-un tip în altul

4.2.4. Funcții de conversie dintr-un tip în altul

Aceste funcții au ca argumente date de tip **DATE**, **NUMBER**, **CHAR** și returnează date de tip **DATE**, **NUMBER**, **CHAR**.

4.2.4. Funcții de conversie dintr-un tip în altul

Funcțiile de conversie din **Oracle** se pot folosi pentru a converti diverse formate:

- a) Conversia din dată calendaristică în șir de caractere
- b) Conversia din șir de caractere în dată calendaristică
- c) Conversia din număr în șir de caractere
- d) Conversia din șir de caractere în număr

4.2.4. Funcții de conversie dintr-un tip în altul

a) Conversia din dată calendaristică în șir de caractere

Conversia unei date calendaristice în șir de caractere se poate realiza cu ajutorul funcției **TO_CHAR**.

4.2.4. Funcții de conversie dintr-un tip în altul

a) Conversia din dată calendaristică în șir de caractere

Sintaxa acestei funcții este: **TO_CHAR(dt, format)**

dt poate avea unul din tipurile pentru date calendaristice:

1. DATE, TIMESTAMP
2. TIMESTAMP WITH TIME ZONE
3. TIMESTAMP
4. WITH LOCAL TIME ZONE
5. INTERVAL MONTH TO YEAR
6. INTERVAL DAY TO SECOND

4.2.4. Funcții de conversie dintr-un tip în altul

Formatul poate conține mai mulți parametrii care pot afecta modul în care va arăta șirul returnat.

Câțiva din acești parametri:

Forma	Parametru	Descriere	Exemplu
Secolul	CC	Secolul scris cu două cifre	21
Trimestrul	Q	Trimestrul din an în care se găsește data	2
Anul	YYYY, RRRR	Anul scris cu patru cifre	2023
	YY, RR	Ultimele două cifre din an	23
	Y	Ultima cifră din an	3
	YEAR, Year	Numele anului	TWO THOUSAND TWENTY-THREE, two thousand twenty-three

4.2.4. Funcții de conversie dintr-un tip în altul

Forma	Parametru	Descriere	Exemplu
Luna	MM	Luna cu două cifre	03
	MONTH, Month	Numele complet al lunii	MARCH, March
	MON, Mon	Primele trei litere ale denumirii lunii	MAR, Mar
	RM	Luna scrisă cu cifre romane	III
Săptămâna	WW	Numărul săptămânii din an	14
	W	Ultima cifră a numărului săptămânii din an	2

4.2.4. Funcții de conversie dintr-un tip în altul

Forma	Parametru	Descriere	Exemplu
Ziua	DDD	Numărul zilei din cadrul anului	57
	DD	Numărul zilei în cadrul lunii	31
	D	Numărul zilei în cadrul săptămânii	5
	DAY, Day	Numele complet al zilei din săptămână	SATURDAY, Saturday
	DY, Dy	Prescurtarea denumirii zilei din săptămână	SAT, Sat
Ora	HH24	Ora în formatul cu 24 de ore	23
	HH	Ora în formatul cu 12 ore	11
Minute	MI	Minutele cu două cifre	68
Secunde	SS	Secundele cu două cifre	34
Sufixe	AM sau PM	AM sau PM după cum e cazul	AM

4.2.4. Funcții de conversie dintr-un tip în altul

În cadrul formatului se pot folosi oricare dintre următorii separatori - /, ., :, :

Dacă în șirul returnat dorim să includem și anumite texte acestea se vor scrie între ghilimele.

Exemplul 1:

```
select sysdate, to_char(sysdate,'MONTH DD, YYYY'),  
       to_char(sysdate,'Month DD, YYYY'), to_char(sysdate,'Mon DD,  
       YYYY') from dual
```

```
1 select sysdate, to_char(sysdate,'MONTH DD, YYYY'), to_char(sysdate,'Month DD, YYYY'), to_char(sysdate,'Mon DD, YYYY')  
2 from dual  
3
```

Results Explain Describe Saved SQL History

SYSDATE	TO_CHAR(SYSDATE,'MONTHDD,YYYY')	TO_CHAR(SYSDATE,'MONTHDD,YYYY')	TO_CHAR(SYSDATE,'MONDD,YYYY')
03/09/2023	MARCH 09, 2023	March 09, 2023	Mar 09, 2023

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 2:

```
select to_char(sysdate, "Trimestrul "Q "al anului  
" Year")  
from dual
```

```
1 select to_char(sysdate, 'Trimestrul "Q "al anului " Year')  
2 from dual  
3
```

Results	Explain	Describe	Saved SQL	History
TO_CHAR(SYSDATE, 'TRIMESTRUL"Q"ALANULUI"YEAR')				
Trimestrul 1 al anului Twenty Twenty-Three				

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 3:

```
select to_char(sysdate, 'Secolul "CC')  
from dual
```

```
1  select to_char(sysdate, 'Secolul "CC')  
2  from dual  
3  
4
```

Results	Explain	Describe	Saved SQL	History
TO_CHAR(SYSDATE, 'SECOLUL"CC')				
Secolul 21				

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 4:

```
select to_char(sysdate,'Day, dd.RM.YYYY')  
from dual
```

```
1  select to_char(sysdate,'Day, dd.RM.YYYY')  
2  from dual  
3  |
```

Results	Explain	Describe	Saved SQL	History
TO_CHAR(SYSDATE,'DAY,DD.RM.YYYY')				
Thursday , 09.III .2023				

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 5:

```
select to_char(sysdate,'Dy, D, DD, DDD'), sysdate  
from dual
```

```
1  select to_char(sysdate,'Dy, D, DD, DDD'), sysdate  
2  from dual  
3
```

Results

Explain

Describe

Saved SQL

History

TO_CHAR(SYSDATE,'DY,D,DD,DDD')**SYSDATE**

Thu, 5, 09, 068

03/09/2023

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 6:

```
select to_char(sysdate,'HH24:MI/HH:MI AM')  
from dual
```

```
1  select to_char(sysdate,'HH24:MI/HH:MI AM')  
2  from dual  
3
```

Results	Explain	Describe	Saved SQL	History
TO_CHAR(SYSDATE,'HH24:MI/HH:MIAM')				
15:44/03:44 PM				

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 7:

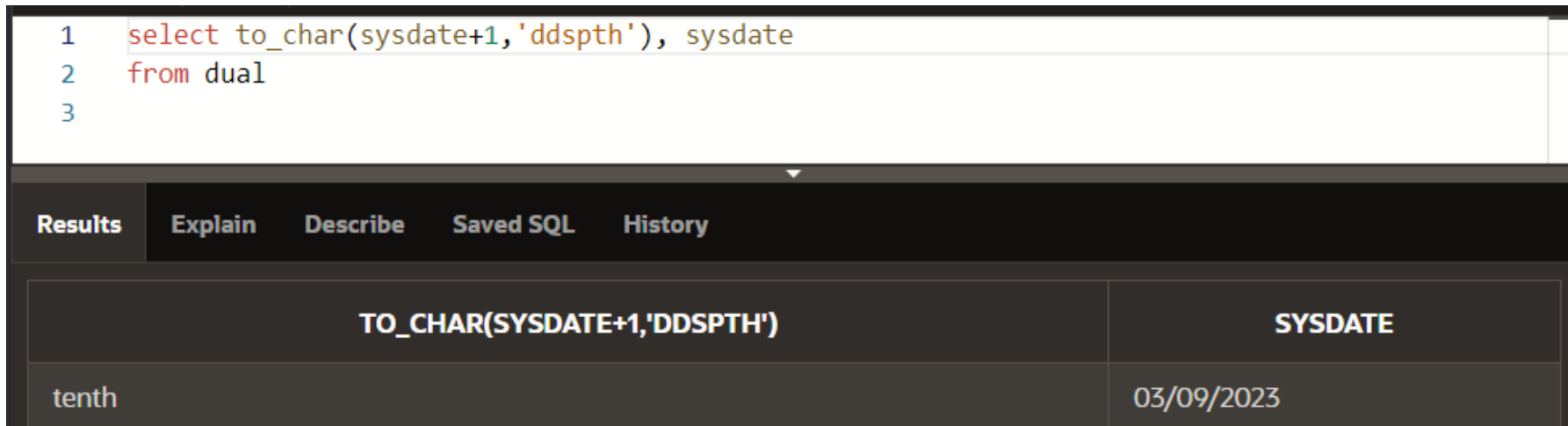
```
select to_char(sysdate+1,'ddth'), sysdate  
from dual
```

<pre>1 select to_char(sysdate+1,'ddth'), sysdate 2 from dual 3 </pre>	
Results Explain Describe Saved SQL History	
TO_CHAR(SYSDATE+1,'DDTH')	SYSDATE
10th	03/09/2023

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 8:

```
select to_char(sysdate+1,'ddspth'), sysdate  
from dual
```



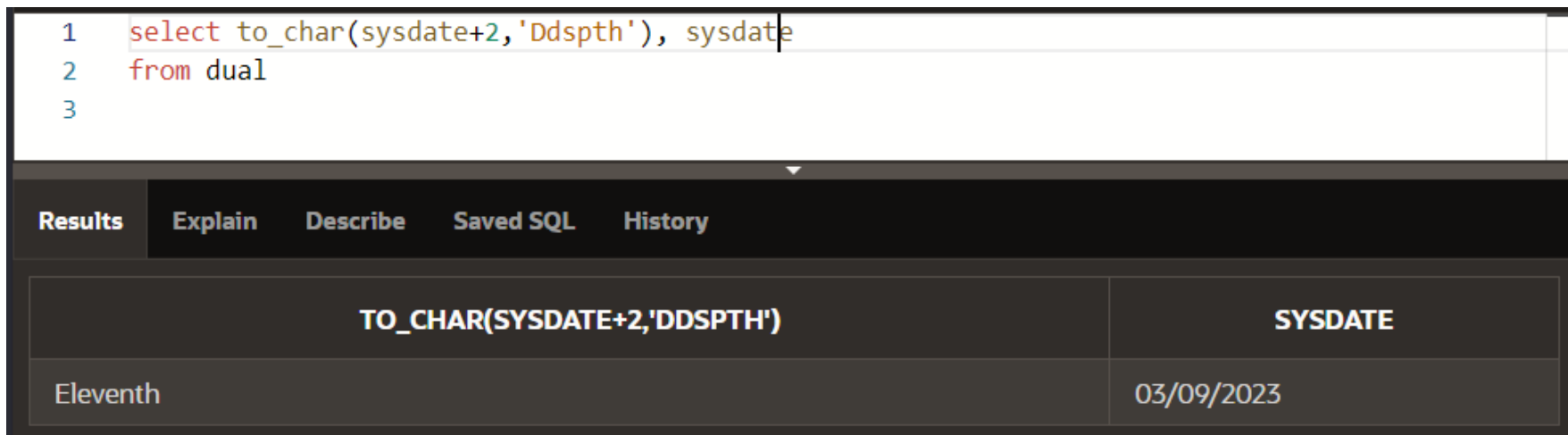
The screenshot shows a SQL query execution interface. The query is: `select to_char(sysdate+1,'ddspth'), sysdate from dual`. The results are displayed in a table with two columns: `TO_CHAR(SYSDATE+1,'DDSPH')` and `SYSDATE`. The first column contains the value 'tenth' and the second column contains the date '03/09/2023'.

Results Explain Describe Saved SQL History				
TO_CHAR(SYSDATE+1,'DDSPH')		SYSDATE		
tenth		03/09/2023		

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 9:

```
select to_char(sysdate+2,'Ddspth'), sysdate  
from dual
```



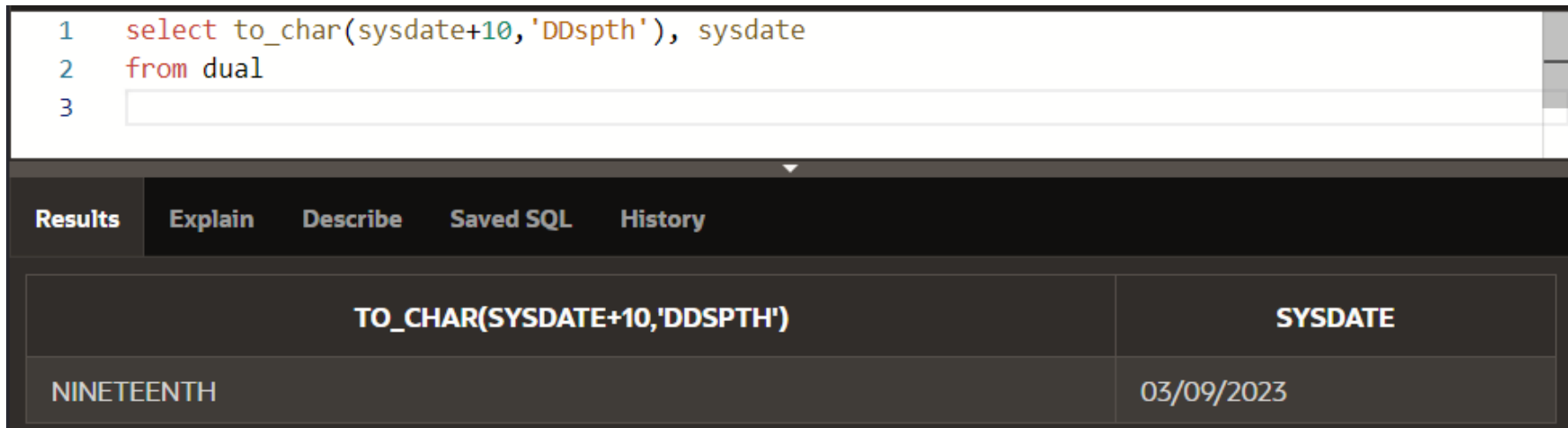
The screenshot shows a SQL query execution interface. The query is: `select to_char(sysdate+2,'Ddspth'), sysdate from dual`. The results are displayed in a table with two columns: `TO_CHAR(SYSDATE+2,'DDSPTH')` and `SYSDATE`. The first row of results shows the value `Eleventh` for the first column and `03/09/2023` for the second column. The interface includes tabs for `Results`, `Explain`, `Describe`, `Saved SQL`, and `History`.

TO_CHAR(SYSDATE+2,'DDSPTH')	SYSDATE
Eleventh	03/09/2023

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 10:

```
select to_char(sysdate+10,'DDspth'), sysdate  
from dual
```



The screenshot shows a SQL query execution interface. The query is: `select to_char(sysdate+10,'DDspth'), sysdate from dual`. The results are displayed in a table with two columns: `TO_CHAR(SYSDATE+10,'DDSPTH')` and `SYSDATE`. The first row shows the values `NINETEENTH` and `03/09/2023`.

TO_CHAR(SYSDATE+10,'DDSPTH')	SYSDATE
NINETEENTH	03/09/2023

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 11:

```
select to_char(sysdate,'mmss'), sysdate  
from dual
```

```
1  select to_char(sysdate,'mmss'), sysdate  
2  from dual  
3  |
```

Results

Explain

Describe

Saved SQL

History

TO_CHAR(SYSDATE,'MMSS')

SYSDATE

three

03/09/2023

4.2.4. Funcții de conversie dintr-un tip în altul

b) Conversia din șir de caractere în dată calendaristică

Folosind funcția **TO_DATE** se poate transforma un șir de caractere precum 'Match 09, 2023' într-o dată calendaristică.

Sintaxa funcției este: **TO_DATE(sir, format)**

Formatul nu este obligatoriu, însă dacă nu este precizat, șirul trebuie să respecte formatul implicit al datei calendaristice DD-MON-YYYY sau DD-MON-YY. Formatul poate folosi aceiași parametrii de format ca și funcția **TO_CHAR**.

4.2.4. Funcții de conversie dintr-un tip în altul

Exemple:

```
select to_date('3.21.23', 'MM/DD/YY')  
from dual;
```

```
1  select to_date('3.21.23', 'MM/DD/YY')  
2  from dual;  
3  |
```

Results

Explain

Describe

Saved SQL

History

TO_DATE('3.21.23','MM/DD/YY')

03/21/2023

4.2.4. Funcții de conversie dintr-un tip în altul

Exemple:

```
select to_date('010123','ddmmyy')  
from dual
```

```
1 select to_date('010123','ddmmyy')  
2 from dual  
3 |
```

Results

Explain

Describe

Saved SQL

History

TO_DATE('010123','DDMMYY')

01/01/2023

4.2.4. Funcții de conversie dintr-un tip în altul

Formatele **RR** si **YY**

Stim ca în formatarea unei date calendaristice se pot folosi pentru an atât **YY** cât și **RR**.

Diferența dintre aceste două formate este modul în care ele interpretează anii aparținând de secole diferite.

Oracle memorează toate cele patru cifre ale unui an, dar dacă sunt transmise doar două din aceste cifre, **Oracle** va interpreta secolul diferit în cazul celor două formate.

4.2.4. Funcții de conversie dintr-un tip în altul

Formatele **RR** si **YY**

```
select to_char(to_date('09-MAR-95','DD-MON-YY'),  
'DD-MON-YYYY') as "YY Format", to_char(to_date('09-  
MAR-95','DD-MON-RR'), 'DD-MON-RRRR') as "RR  
Format"  
from dual
```



The screenshot shows a SQL query execution interface. The query is as follows:

```
1 select to_char(to_date('09-MAR-95','DD-MON-YY'), 'DD-MON-YYYY') as "YY Format", to_char(to_date('09-MAR-95','DD-MON-RR'), 'DD-MON-RRRR') as "RR Format"  
2 from dual  
3
```

Below the query, there is a tabbed interface with 'Results' selected. The results are displayed in a table with two columns: 'YY Format' and 'RR Format'.

YY Format	RR Format
09-MAR-2095	09-MAR-1995

4.2.4. Funcții de conversie dintr-un tip în altul

Formatele RR si YY

Se observă modul diferit de interpretare a anului.
Dacă utilizați formatul YY și anul este specificat doar prin două cifre, se presupune că anul respectiv face parte din același secol cu anul curent

```
1 select to_char(to_date('09-MAR-95','DD-MON-YY'),'DD-MON-YYYY') as "YY Format", to_char(to_date('09-MAR-95','DD-MON-RR'),'DD-MON-RRRR') as "RR Format"
2 from dual
3
```

YY Format	RR Format
09-MAR-2095	09-MAR-1995

4.2.4. Funcții de conversie dintr-un tip în altul

c) Conversia din număr în șir de caractere

Pentru a transforma un număr într-un șir de caractere, se folosește funcția **TO_CHAR**, cu următoarea sintaxă:

TO_CHAR(numar, format)

format - poate conține unul sau mai mulți parametri de formatare dintre cei prezentați în tabelul următor

4.2.4. Funcții de conversie dintr-un tip în altul

Parametru format	Exemplu	Descriere
9	999	Returnează cifrele numărului din pozițiile specificate, precedat de semnul minus dacă numărul este negativ
0	0999	Completează cifrele numărului cu zerouri în față
.	999.99	Specifică poziția punctului zecimal
,	9,999	Specifică poziția separatorului virgulă
\$	\$999	Afișează semnul dolar
EEEE	9.99EEEE	Returnează scrierea științifică a numărului

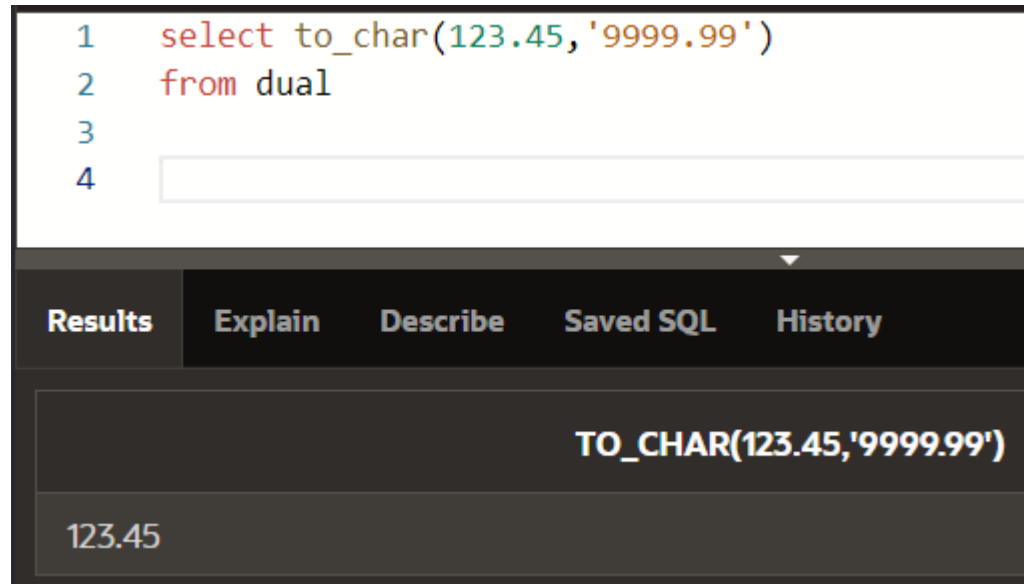
4.2.4. Funcții de conversie dintr-un tip în altul

Parametru format	Exemplu	Descriere
L	L999	Afișează simbolul monetar
MI	999MI	Afișează semnul minus după număr dacă acesta este negativ
PR	999PR	Numerele negative sunt închise între paranteze unghiulare
RN rn	RN Rn	Afișează numărul în cifre romane
V	99V99	Afișează numărul înmulțit cu 10 la puterea x, și rotunjit la ultima cifră, unde x este numărul de cifre 9 de după V
X	XXXX	Afișează numărul în baza 16

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 1:

```
select to_char(123.45,'9999.99')  
from dual
```



The screenshot shows a SQL query execution interface. The query is: `select to_char(123.45,'9999.99')` from dual. The results are displayed in a table with the column header `TO_CHAR(123.45,'9999.99')` and the value `123.45`.

TO_CHAR(123.45,'9999.99')
123.45

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 2:

```
select to_char(123.45,'0000.000')  
from dual
```

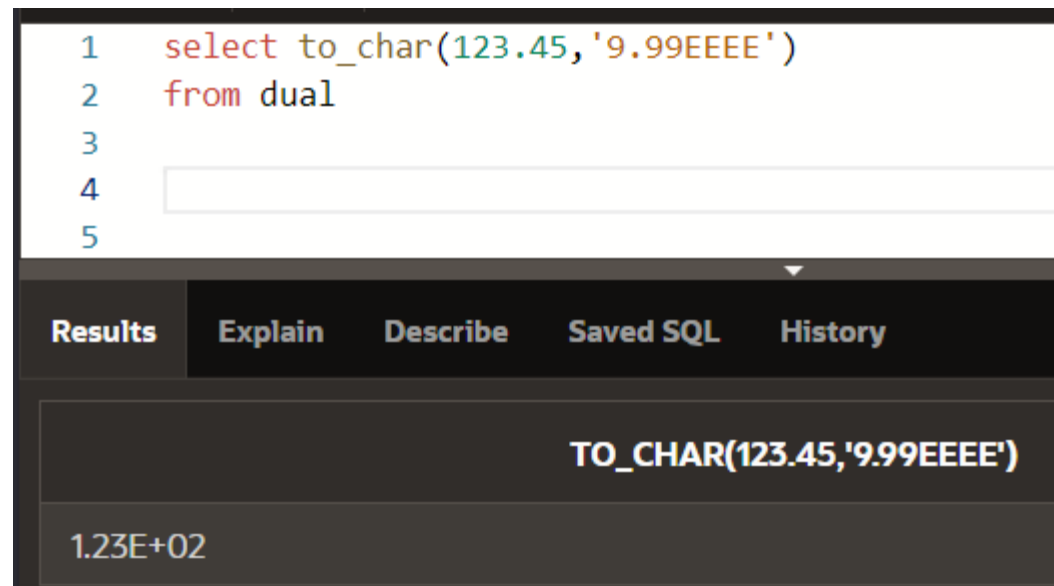
```
1  select to_char(123.45,'0000.000')  
2  from dual  
3  
4  |  
5
```

Results	Explain	Describe	Saved SQL	History
TO_CHAR(123.45,'0000.000')				
0123.450				

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 3:

```
select to_char(123.45,'9.99EEEE')  
from dual
```



The screenshot shows a SQL query execution interface. The query is: `select to_char(123.45,'9.99EEEE') from dual`. The results are displayed in a table with one row and one column. The column header is `TO_CHAR(123.45,'9.99EEEE')` and the value is `1.23E+02`.

TO_CHAR(123.45,'9.99EEEE')
1.23E+02

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 4:

```
select to_char(-123.45,'999.999PR')  
from dual
```

The screenshot shows a SQL query execution interface. The query is: `select to_char(-123.45,'999.999PR') from dual`. The results are displayed in a table with one row and one column. The column header is `TO_CHAR(-123.45,'999.999PR')` and the value is `<123.450>`.

TO_CHAR(-123.45,'999.999PR')
<123.450>

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 5:

```
select to_char(1.2373,'99999V99')  
from dual
```

The screenshot shows a SQL query execution interface. The query is: `select to_char(1.2373,'99999V99') from dual`. The interface has a tabbed view with 'Results' selected. The result is displayed in a table with one row and one column. The column header is `TO_CHAR(1.2373,'99999V99')` and the value is `124`.

TO_CHAR(1.2373,'99999V99')
124

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 6:

```
select to_char(1.2373,'L0000.000')  
from dual
```

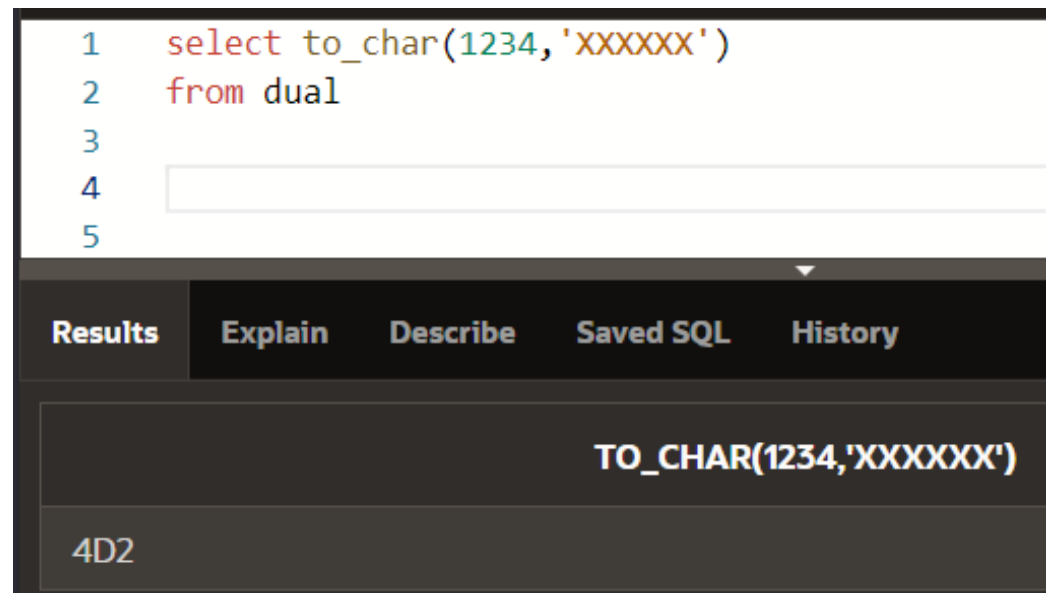
The screenshot shows a SQL query execution interface. The query is: `select to_char(1.2373,'L0000.000') from dual`. The interface has a tabbed view with 'Results' selected. The result is displayed in a table with one column, `TO_CHAR(1.2373,'L0000.000')`, and one row, `$0001.237`.

TO_CHAR(1.2373,'L0000.000')
\$0001.237

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 7:

```
select to_char(1234,'XXXXXX')  
from dual
```



The screenshot shows a SQL query execution interface. The query is: `select to_char(1234,'XXXXXX') from dual`. The interface has a tabbed view with 'Results' selected. The result is displayed in a table with one column named `TO_CHAR(1234,'XXXXXX')` and one row containing the value `4D2`.

TO_CHAR(1234,'XXXXXX')
4D2

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplul 8:

```
select to_char(987,'RN')  
from dual
```

The screenshot shows a SQL query execution interface. The query is: `select to_char(987,'RN') from dual`. The interface has a tabbed view with 'Results' selected. The result is displayed in a table with one column named `TO_CHAR(987,'RN')` and one row containing the value `CMLXXXVII`.

```
1  select to_char(987,'RN')  
2  from dual  
3  
4  
5
```

TO_CHAR(987,'RN')
CMLXXXVII

4.2.4. Funcții de conversie dintr-un tip în altul

d) Conversia din șir de caractere în număr

Transformarea inversă din șir de caractere într-o valoare numerică se realizează cu ajutorul funcției

TO_NUMBER:

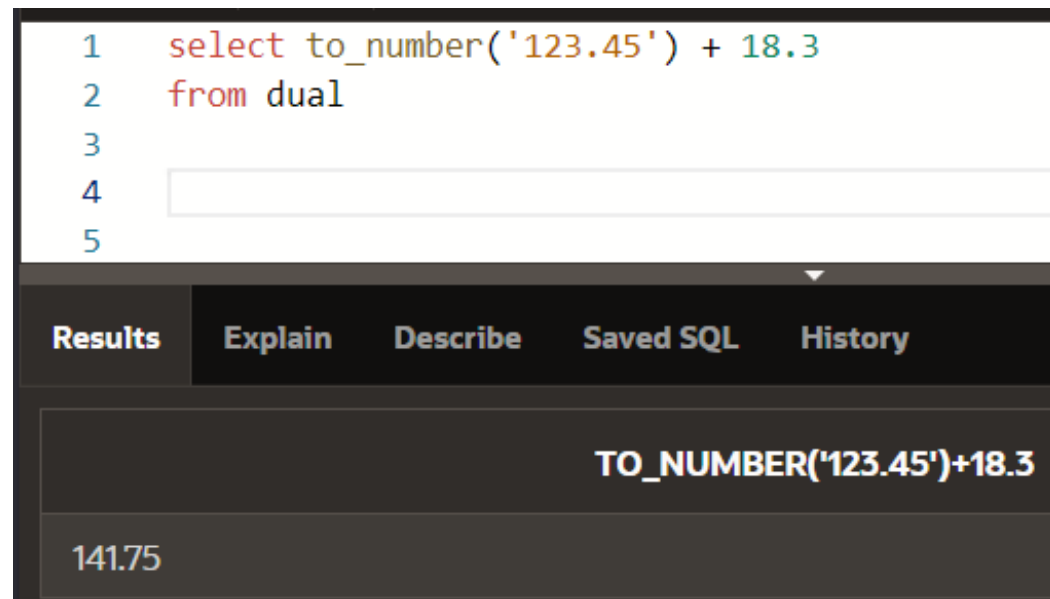
TO_NUMBER(sir, format)

Parametrii de formatare a sirului ce se pot folosi sunt aceiași ca în cazul funcției **TO_CHAR**

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplu 1:

```
select to_number('123.45') + 18.3  
from dual
```



The screenshot shows a SQL query execution interface. The query is entered in a text area and is as follows:

```
1 select to_number('123.45') + 18.3  
2 from dual  
3  
4  
5
```

Below the query area, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, and it displays the result of the query in a table with one row and one column:

TO_NUMBER('123.45')+18.3
141.75

4.2.4. Funcții de conversie dintr-un tip în altul

Exemplu 2:

```
select to_number('-98,765.43','$99,999.99')  
from dual;
```

```
1  select to_number('-98,765.43','$99,999.99')  
2  from dual;  
3  
4  
5
```

Results	Explain	Describe	Saved SQL	History
TO_NUMBER('-98,765.43','\$99,999.99')				
-98765.43				

Limbajul SQL

Interogări SELECT pe o singură tabelă (partea II)

4.2. Funcții referitoare la o singură înregistrare

4.2.1. Funcții pentru șiruri de caractere

4.2.2. Funcții de tip numeric

4.2.3. Funcții de tip dată calendaristică și oră

4.2.4. Funcții de conversie dintr-un tip în altul

4.2.5. Funcții generale

4.2.6. Funcții condiționale

4.2.5. Funcții generale

Funcții generale:

1. NVL
2. NVL2
3. NULLIF
4. COALESCE

Aceste funcții au ca argumente date de diferite tipuri și returnează date de tipuri diferite.

Aceste funcții precizează cum sunt prelucrate valorile de tip **NULL**.

4.2.5. Funcții generale

1. Funcția **NVL** cu formatul:

NVL(valoare1, valoare2)

Returneaza **valoare1**, daca este nenula, sau returneaza **valoare2**, daca valoare1 este **NULL**.

Funcția prelucreaza date de tipurile caracter, numeric sau data calendaristica, cu precizarea ca ambele valori parametru sunt de acelasi tip.

4.2.5. Funcții generale

Exemplu:

```
select ename, comm, NVL(comm,0.8)
```

from emp

where empno between 7600 and 7800

```
1 select ename, comm, NVL(comm,0.8)
2 from emp
3 where empno between 7600 and 7800
4
5
```

Results	Explain	Describe	Saved SQL	History
ENAME	COMM	NVL(COMM,0.8)		
MARTIN	1400	1400		
BLAKE	-	.8		
CLARK	-	.8		
SCOTT	-	.8		

4.2.5. Funcții generale

2. Funcția **NVL2** cu formatul:

NVL2(valoare1, valoare2, valoare3)

returneaza **valoare2**, daca **valoare1** este nenula,
iar daca **valoare1** este NULL, atunci returneaza
valoare3.

4.2.5. Funcții generale

Exemplu:

```
select ename, comm, NVL2(comm,'ARE  
    COMISION','NU ARE COMISION')  
from emp  
where empno between 7600 and 7800
```

4.2.5. Funcții generale

Rezultat afisat:

```
1  select ename, comm, NVL2(comm, 'ARE COMISION', 'NU ARE COMISION')
2  from emp
3  where empno between 7600 and 7800
4
5
```

Results

Explain

Describe

Saved SQL

History

ENAME	COMM	NVL2(COMM,'ARECOMISION','NUARECOMISION')
MARTIN	1400	ARE COMISION
BLAKE	-	NU ARE COMISION
CLARK	-	NU ARE COMISION
SCOTT	-	NU ARE COMISION

4.2.5. Funcții generale

3. Funcția **NULLIF** cu formatul:

NULLIF(expresie1, expresie2)

Returneaza **NULL**, daca cele doua expresii sunt egale.

Daca cele doua expresii sunt diferite (valorile lor), atunci returneaza valoarea primei expresii – **expresie1**.

4.2.5. Funcții generale

Exemplu:

```
select empno, ename, job,  
       NULLIF(length(ename),length(job))  
from emp  
where empno between 7300 and 7700
```

4.2.5. Funcții generale

Rezultat afisat:

```
1  select empno, ename, job, NULLIF(length(ename),length(job))
2  from emp
3  where empno between 7300 and 7700
4
5
```

Results	Explain	Describe	Saved SQL	History
EMPNO	ENAME	JOB	NULLIF(LENGTH(ENAME),LENGTH(JOB))	
7369	SMITH	CLERK	-	
7499	ALLEN	SALESMAN	5	
7521	WARD	SALESMAN	4	
7566	JONES	MANAGER	5	
7654	MARTIN	SALESMAN	6	
7698	BLAKE	MANAGER	5	

4.2.5. Funcții generale

4. Funcția **COALESCE** cu formatul:

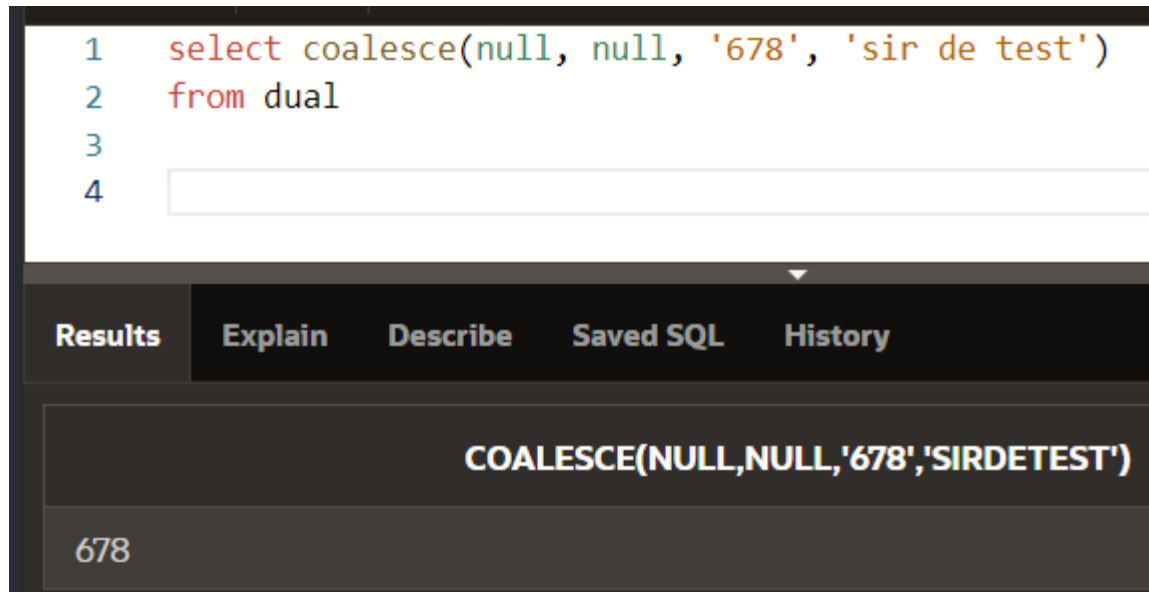
COALESCE(expresie1, expresie2, ..., expresien)

Returneaza valoarea primei expresii nenule.

4.2.5. Funcții generale

Exemplu:

```
select coalesce(null, null, '678', 'sir de test')  
from dual
```



The screenshot shows a SQL query execution interface. The query is:

```
1 select coalesce(null, null, '678', 'sir de test')  
2 from dual  
3  
4
```

Below the query, there are tabs for **Results**, **Explain**, **Describe**, **Saved SQL**, and **History**. The **Results** tab is selected, showing the following result:

COALESCE(NULL,NULL,'678','SIRDETEST')
678

Limbajul SQL

Interogări SELECT pe o singură tabelă (partea II)

4.2. Funcții referitoare la o singură înregistrare

4.2.1. Funcții pentru șiruri de caractere

4.2.2. Funcții de tip numeric

4.2.3. Funcții de tip dată calendaristică și oră

4.2.4. Funcții de conversie dintr-un tip în altul

4.2.5. Funcții generale

4.2.6. Funcții condiționale

4.2.5. Funcții generale

- SGBD-ul **ORACLE** pune la dispoziția programatorilor, în cadrul limbajului **SQL**, o funcție și o expresie condițională.
- Acestea sunt alternative foarte bune la structurile de tip **IF-THEN-ELSE**.
- Funcția se numește **DECODE**, iar expresia condițională este **CASE**.

4.2.5. Funcții generale

Funcția **DECODE** cu formatul:

**DECODE(expresie, valoare1_1, valoare1_2,
valoare2_1, valoare2_2, ..., valoaren_1,
valoaren_2, valoare)**

Compara valoarea **expresiei** cu fiecare din **valoare1_1, valoare2_1, ..., valoaren_1**. Dacă valoarea **expresie** este egală cu **valoarei_1**, atunci va returna **valoarei_2**. Dacă nici una din valorile **valoare1_1, valoare2_1, ..., valoaren_1** nu este egală cu **expresie**, atunci va returna **valoare**.

4.2.5. Funcții generale

Exemplu 1:

```
select DECODE('SQL' , 'PLSQL', 'Limbajul PLSQL',  
             'SQL', 'Limbajul de interogare SQL', 'Limbaj de  
             programare')  
from dual
```

```
1  select DECODE('SQL' , 'PLSQL', 'Limbajul PLSQL', 'SQL', 'Limbajul de interogare SQL', 'Limbaj de programare'  
2  from dual  
3  
4
```

Results

Explain

Describe

Saved SQL

History

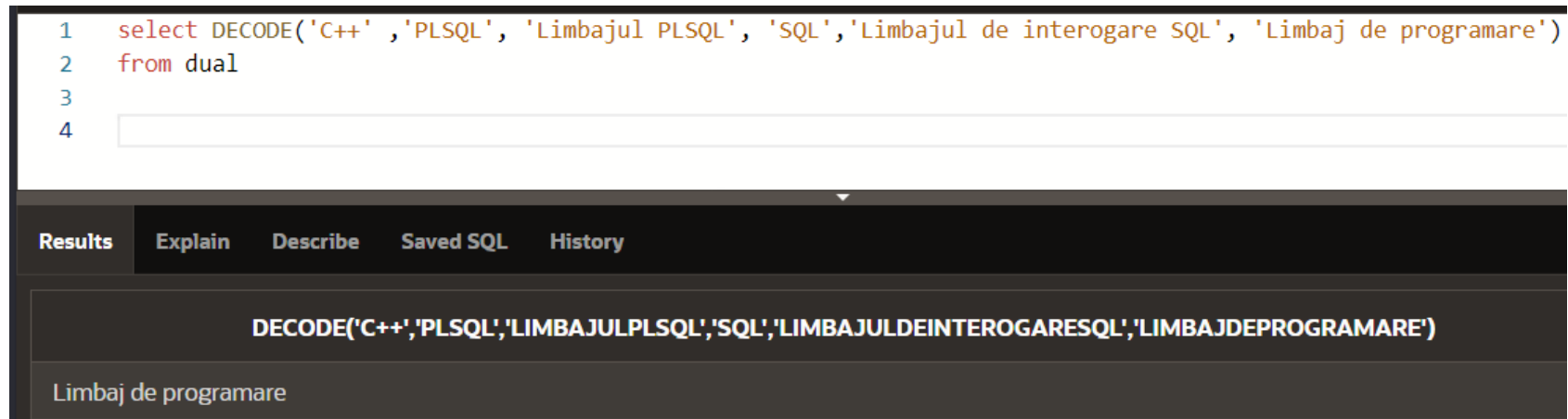
DECODE('SQL','PLSQL','LIMBAJULPLSQL','SQL','LIMBAJULDEINTEROGARESOL','LIMBAJDEPROGRAMARE')

Limbajul de interogare SQL

4.2.5. Funcții generale

Exemplu 2:

```
select DECODE('C++' , 'PLSQL', 'Limbajul PLSQL',  
              'SQL', 'Limbajul de interogare SQL', 'Limbaj de  
              programare')  
from dual
```



The screenshot shows a SQL query execution interface. The query is:

```
1 select DECODE('C++' , 'PLSQL', 'Limbajul PLSQL', 'SQL', 'Limbajul de interogare SQL', 'Limbaj de programare')  
2 from dual  
3  
4
```

The interface has tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing the following result:

DECODE('C++','PLSQL','LIMBAJULPLSQL','SQL','LIMBAJULDEINTEROGARESOL','LIMBAJDEPROGRAMARE')
Limbaj de programare

4.2.5. Funcții generale

Expresia condicionala **CASE** are urmatorul format:

CASE expresie

WHEN valoare1_1 THEN valoare1_2

WHEN valoare2_1 THEN valoare2_2

...

WHEN valoaren_1 THEN valoaren_2

ELSE valoare

END

4.2.5. Funcții generale

- Expresia condicională **CASE** folosește cuvinte cheie **WHEN**, **THEN**, **ELSE** și **END**. Ca și regula generală orice expresie care poate fi scrisă cu ajutorul funcției **DECODE**, poate fi transcrisă și cu ajutorul expresiei conditionale **CASE**.
- Folosind expresia condicională **CASE** obținem un cod mai lung, dar mai ușor de înțeles și de depanat.

4.2.5. Funcții generale

Exemplu 1:

```
select CASE 'PLSQL'  
  WHEN 'PLSQL' THEN 'Limbajul PLSQL'  
  WHEN 'SQL' THEN 'Limbajul de interogare SQL'  
  ELSE 'Limbaj de programare'  
END  
from dual
```

```
1  select CASE 'PLSQL'  
2    WHEN 'PLSQL' THEN 'Limbajul PLSQL'  
3    WHEN 'SQL' THEN 'Limbajul de interogare SQL'  
4    ELSE 'Limbaj de programare'  
5  END  
6  from dual  
7  
8
```

Results Explain Describe Saved SQL History

CASE'PLSQL'WHEN'PLSQL'THEN'LIMBAJULPLSQL'WHEN'SQL'THEN'LIMBAJULDEINTEROGARESOL'ELSE'LIMBAJDEPROGRAMARE'END

Limbajul PLSQL

4.2.5. Funcții generale

Exemplu 2:

```
select CASE 'C++'  
  WHEN 'PLSQL' THEN 'Limbajul PLSQL'  
  WHEN 'SQL' THEN 'Limbajul de interogare SQL'  
  ELSE 'Limbaj de programare'  
END  
from dual
```

```
1  select CASE 'C++'  
2    WHEN 'PLSQL' THEN 'Limbajul PLSQL'  
3    WHEN 'SQL' THEN 'Limbajul de interogare SQL'  
4    ELSE 'Limbaj de programare'  
5    END  
6  from dual  
7  |  
8
```

Results Explain Describe Saved SQL History

CASE'C++'WHEN'PLSQL'THEN'LIMBAJULPLSQL'WHEN'SQL'THEN'LIMBAJULDEINTEROGARESQL'ELSE'LIMBAJDEPROGRAMARE'END

Limbaj de programare

Referințe bibliografice

- 1) <https://docs.oracle.com/cloud/help/ro/analytics-cloud/ACUBI/GUID-4CBCE8D4-CF17-43BD-AAEF-C5D614A8040A.htm#BILUG672>
- 2) https://www.tutorialspoint.com/sql_certificate/using_single_row_functions.htm
- 3) <https://www.w3resource.com/sql-exercises/>

Întrebări?