

# BAZE DE DATE - 3

## Autor

id_autor	nume_autor
1	Petru
2	Diuma
3	Vieru
4	Lena
5	Ioana
6	Ghita
7	Colea
8	Nicu
9	Valea

## Carti

idcarte	autor	titlu	pret	cantitatea	id_autor
2-2222-222-10	Petru	GFGF Laborator Mysql-Php vbvbvb	950.00	23	1
2-2222-222-13	Vieru	MAMA	2000.00	200	3
2-2222-2222-22	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-23	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-6	Vica	Cei trei muschetari Php	930.00	40	4
2-2222-2222-8	Eminescu	Ghid Php	1000.00	3	6
2-2222-2222-9	Eminescu	Ghid Php	1000.00	3	2

# Limbajul SQL

## Cereri SELECT pe o tabela

1. **Funcții**
2. Funcții referitoare la o singură înregistrare
3. Funcții referitoare la mai multe înregistrări
  1. Clauza **GROUP BY**
  2. Excluderea grupurilor (clauza **HAVING**)
  3. Imbricarea funcțiilor de grup

# Tabele Angajati si Departamente

Pentru exemplele din cursuri vom folosi tabelele ***Angajati*** si ***Departamente***.

## DEPARTMENT

Id\_dept **NUMBER(3)** CHEIE PRIMARA (**PK**)

Den\_dept **VARCHAR2(20)**

Id\_manager **VARCHAR2(3)**

Locatie **VARCHAR2(100)**

```
insert into
Departament(Id
er,Locatie) VA
('1','IT','1','Chis
('2','Exploatare
('3','Managem
('4','HR','3','Ca
('5','Contabilita
('6','Software',
```

```
Create Table Departament(Id_dept
int(3) not null auto_increment
primary key,
Den_dept VARCHAR(20),
Id_manager VARCHAR(3),
Locatie VARCHAR(50)
);
```

# Tabele Angajati si Departamente

## ANGAJATI

Id\_angajat **NUMBER(3)** CHEIE PRIMARA (**PK**)

Id\_dept **NUMBER(3)** REFERINTA (**FK**) LA TABELA  
**DEPARTAMENTE**

Nume **VARCHAR (40)**

Prenume **VARCHAR (40)**

Functie **VARCHAR (25)**

Salariu **NUMBER(7)**

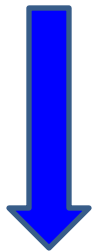
Id\_manager **VARCHAR (3)**

Data\_ang **DATE**

Comision **NUMBER(5)**

Id_angajat	Id_dept	Nume	Prenume
1	1	Petrov	Petru
2	1	Ivanov	Ion
3	2	Volodea	Petru
4	3	Rebricov	Ghena
5	4	Vasiliev	Ion

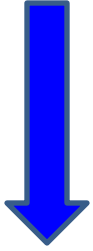
```
create table ANGAJATI (  
  Id_angajat int(3) not null auto_increment  
  primary key,  
  Id_dept int(3) not null,  
  Nume VARCHAR(40)not null,  
  Prenume VARCHAR(40)not null,  
  Functie VARCHAR(25)not null,  
  Salariu real(9,2)not null,  
  Id_manager VARCHAR(3)not null,  
  Data_ang DATE not null,  
  Comision int(5)not null,  
  FOREIGN KEY (Id_dept) REFERENCES  
  Departament(Id_dept)  
);
```



# Funcții

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

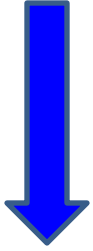
- 1. a realiza calcule asupra datelor**
- 2. a modifica date**
- 3. a manipula grupuri de înregistrări**
- 4. a schimba formatul datelor**
- 5. sau pentru a converti diferite tipuri de date**



# Funcții

Funcțiile se clasifică în două tipuri:

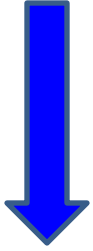
1. **Funcții referitoare la o singură înregistrare (single-row functions)**
2. **Funcții referitoare la mai multe înregistrări (multiple-row functions)**



# Funcții

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

1. funcții caracter
2. funcții numerice
3. funcții pentru data calendaristică si oră
4. funcții de conversie
5. funcții diverse

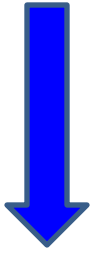


# Funcții

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

- funcții totalizatoare sau funcții de grup

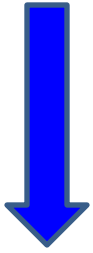




# Funcții

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.*

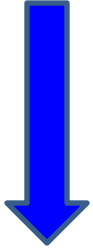


# Funcții

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**

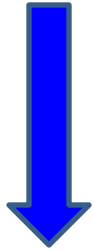


# Limbaajul SQL

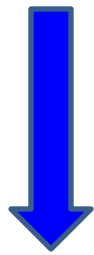
## Cereri SELECT pe o tabela

1. Funcții
2. Funcții referitoare la o singură înregistrare
3. Funcții referitoare la mai multe înregistrări
  - Clauza **GROUP BY**
  - Excluderea grupurilor (clauza **HAVING**)
  - Imbricarea funcțiilor de grup

# Funcții referitoare la o singură înregistrare



- *Sunt funcții utilizate pentru manipularea datelor individuale.*
- Ele pot avea unul sau mai multe argumente și returnează o valoare pentru fiecare rând rezultat în urma interogării.



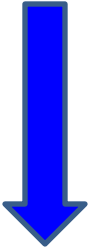
## Funcții referitoare la o singură înregistrare

Sunt mai multe tipuri de funcții pe un singur rând.

Sintaxa:

***function\_name [(arg1,arg2,...)]***

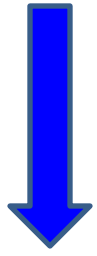
În sintaxa generală ***function\_name*** este numele funcției și ***arg1,arg2*** sunt argumentele funcției care pot fi date de **numele unei coloane sau de o expresie**



# Funcții referitoare la o singură înregistrare

Funcțiile pe un singur rând cuprind următoarele tipuri de funcții:

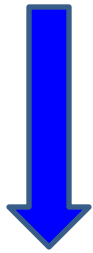
1. funcții de tip caracter
2. funcții de tip numeric
3. funcții de tip data
4. funcții de conversie
5. funcții generale: NVL, NVL2, NULLIF, COALESCE, CASE, DECODE



# Funcții referitoare la o singură înregistrare

## 1. Funcții de tip caracter

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



# Funcții referitoare la o singură înregistrare

Cele mai importante funcții caracter sunt:

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





# Funcții referitoare la o singură înregistrare

Exemplu de utilizare a funcției **LENGTH**:

```
SELECT LENGTH(ume)  
FROM angajati;
```

idcarte	autor	titlu	pret	cantitatea	id_autor
2-2222-222-10	Petru GFGF	Laborator Mysql-Php vbvbvb	950.00	23	1
2-2222-222-13	Vieru	MAMA	2000.00	200	3
2-2222-2222-22	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-23	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-6	Vica	Cei trei muschetari Php	930.00	40	4
2-2222-2222-8	Eminescu	Ghid Php	1000.00	3	6
2-2222-2222-9	Eminescu	Ghid Php	1000.00	3	2

Sa se creeze urmatoarele tabele:

## Tabela DEPARTAMENTE

Id\_dept number(3) cheie primara (PK)

Den\_dept varchar2(20)

Id\_manager varchar2(3)

Locatie varchar2(100)

## Tabela ANGAJATI

Id\_angajat number(3) cheie primara (PK)

Id\_dept number(3) referinta (FK) la tabela **DEPARTAMENTE**

Nume varchar2(40)

Prenume varchar2(40)

Funcție varchar2(25)

Salariu number(7)

Id\_manager varchar2(3)

Data\_ang date

Comision number(5)

id_autor	nume
1	P
2	D
3	V
4	L
5	I
6	G
7	C
8	N
9	V

# Funcții referitoare la o singură înregistrare

Exemplu:

```
SELECT 'Numele functiei pentru  
' || UPPER(nume) || 'este ' || LOWER(functie) AS  
"DETALII ANGAJAT"  
FROM angajati;
```

idcarte	autor	titlu	pret	cantitatea	id_autor
2-2222-222-10	Petru GFGF	Laborator Mysql-Php vbvbvb	950.00	23	1
2-2222-222-13	Vieru	MAMA	2000.00	200	3
2-2222-2222-22	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-23	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-6	Vica	Cei trei muschetari Php	930.00	40	4
2-2222-2222-8	Eminescu	Ghid Php	1000.00	3	6
2-2222-2222-9	Eminescu	Ghid Php	1000.00	3	2

id_autor	nume
1	P
2	D
3	V
4	L
5	I
6	G
7	C
8	N
9	V

Sa se creeze urmatoarele tabele:

## Tabela DEPARTAMENTE

Id\_dept number(3) cheie primara (PK)

Den\_dept varchar2(20)

Id\_manager varchar2(3)

Locatie varchar2(100)

## Tabela ANGAJATI

Id\_angajat number(3) cheie primara (PK)

Id\_dept number(3) referinta (FK) la tabela **DEPARTAMENTE**

Nume varchar2(40)

Prenume varchar2(40)

Functie varchar2(25)

Salariu number(7)

Id\_manager varchar2(3)

Data\_ang date

Comision number(5)

# Funcții referitoare la o singură înregistrare

Exemplu:

```
SELECT id_angajat, UPPER(ume), functie, id_dept
FROM angajati
WHERE nume = 'popa'
```

idcarte	autor	titlu	pret	cantitatea	id_autor
2-2222-222-10	Petru	GFGF Laborator Mysql-Php vbvbvb	950.00	23	1
2-2222-222-13	Vieru	MAMA	2000.00	200	3
2-2222-2222-22	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-23	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-6	Vica	Cei trei muschetari Php	930.00	40	4
2-2222-2222-8	Eminescu	Ghid Php	1000.00	3	6
2-2222-2222-9	Eminescu	Ghid Php	1000.00	3	2

11/10/2021

id_autor
1
2
3
4
5
6
7
8
9

Sa se creeze urmatoarele tabele:

## Tabela DEPARTAMENTE

Id\_dept number(3) cheie primara (PK)

Den\_dept varchar2(20)

Id\_manager varchar2(3)

Locatie varchar2(100)

## Tabela ANGAJATI

Id\_angajat number(3) cheie primara (PK)

Id\_dept number(3) referinta (FK) la tabela **DEPARTAMENTE**

Nume varchar2(40)

Prenume varchar2(40)

Functie varchar2(25)

Salariu number(7)

Id\_manager varchar2(3)

Data\_ang date

Comision number(5)

# Funcții referitoare la o singură înregistrare

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

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

Exemplu:

```
SELECT id_ang, UPPER(nume), functie, id_dept  
FROM angajati  
WHERE INITCAP(nume) = 'Popa'
```

idcarte	autor	titlu	pret	cantitatea
2-2222-222-10	Petru	GFGF Laborator Mysql-Php vbvbvb	950.00	23
2-2222-222-13	Vieru	MAMA	2000.00	200
2-2222-2222-22	Eminescu	Ghid Php	1000.00	3
2-2222-2222-23	Eminescu	Ghid Php	1000.00	3
2-2222-2222-6	Vica	Cei trei muschetari Php	930.00	40
2-2222-2222-8	Eminescu	Ghid Php	1000.00	3
2-2222-2222-9	Eminescu	Ghid Php	1000.00	3

# Funcții referitoare la o singură înregistrare

Exemplu:

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

```
SELECT id_ang, CONCAT(ume, functie), ume,  
       UPPER(ume)  
FROM angajati;
```

idcarte	autor	titlu	pret	cantitatea	id_autor
2-2222-222-10	Petru	GFGF Laborator Mysql-Php vbvbvb	950.00	23	1
2-2222-222-13	Vieru	MAMA	2000.00	200	3
2-2222-2222-22	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-23	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-6	Vica	Cei trei muschetari Php	930.00	40	4
2-2222-2222-8	Eminescu	Ghid Php	1000.00	3	6
2-2222-2222-9	Eminescu	Ghid Php	1000.00	3	2

id_autor	nume
1	Petru
2	Dium
3	Vieru
4	Lena
5	Ioana
6	Ghita
7	Cole
8	Nicu
9	Valea

# Funcții referitoare la o singură înregistrare

*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 **nume** vom folosi interogarea:

# Funcții referitoare la o singură înregistrare

Funcția SQL TRANSLATE() înlocuiește o secvență de caractere dintr-un șir cu o altă secvență de caractere. Funcția înlocuiește un singur caracter la un moment dat.

TRANSLATE(string text, from text, to text)

**SELECT** nume, **LENGTH** (nume) - **LENGTH** (**TRANSLATE**(nume,'DA','D'))  
**FROM** angajati;

Sa se creeze urmatoarele tabele:

## Tabela DEPARTAMENTE

Id\_dept number(3) cheie primara (PK)  
Den\_dept varchar2(20)  
Id\_manager varchar2(3)  
Locatie varchar2(100)

## Tabela ANGAJATI

Id\_angajat number(3) cheie primara (PK)  
Id\_dept number(3) referinta (FK) la tabela **DEPARTAMENTE**  
Nume varchar2(40)  
Prenume varchar2(40)  
Functie varchar2(25)  
Salariu number(7)  
Id\_manager varchar2(3)  
Data\_ang date  
Comision number(5)

Функция SQL TRANSLATE() заменяет последовательность символов в строке другой последовательностью символов. Функция заменяет один символ за раз.	NUME	'A'
	GHEORGHIU	0
	MARIN	1
	GEORGESCU	0
	IONESCU	0
	ALBU	1
	VOINEA	1
	STANESCU	1

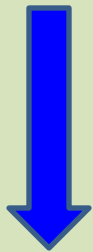
# Funcții referitoare la o singură înregistrare

Notă:

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

## Cereri SELECT pe o tabela

1. Funcții
2. Funcții referitoare la o singură înregistrare
3. Funcții referitoare la mai multe înregistrări
  1. Clauza **GROUP BY**
  2. Excluderea grupurilor (clauza **HAVING**)
  3. Imbricarea funcțiilor de grup



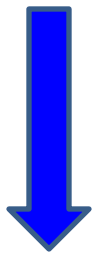
# Funcții de grup

*Funcțiile de grup sunt funcții care operează pe un set de rânduri pentru a da un rezultat pe întreg setul.*

Parametrii și descrierea funcțiilor de grup.

Funcțiile de grup sunt:

1. **AVG**
2. **COUNT**
3. **MAX**
4. **MIN**
5. **STDDEV**
6. **SUM**
7. **VARIANCE**

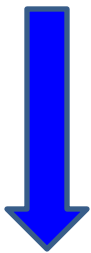


# Funcții de grup

Fiecare dintre aceste funcții acceptă anumiți parametri:

Funcția	Descriere
<b>AVG([DISTINCT ALL] n)</b>	Valoarea medie pentru <b>grup</b> , <b>ignorand valorile nule</b>
<b>COUNT({*  [DISTINCT ALL] expr})</b>	Numarul de randuri unde <b>expr</b> evalueaza <b>altceva in afara de null</b> (folosind * sunt numarate toate randurile, <b>incluzand duplicatele si pe cele cu valoare nula</b> )
<b>MAX([DISTINCT ALL] expr)</b>	Valoarea maxima a <b>expr</b> , <b>ignorand valorile nule</b>
<b>MIN([DISTINCT ALL] expr)</b>	Valoarea minima a <b>expr</b> , <b>ignorand valorile nule</b>
<b>STDDEV([DISTINCT ALL] x)</b>	Deviatia standard pentru grup, <b>ignorand valorile nule</b>
<b>SUM([DISTINCT ALL] x)</b>	Suma valorilor pentru grup, <b>ignorand valorile nule</b>
<b>VARIANCE([DISTINCT ALL] x)</b>	Variatia pentru grup, <b>ignorand valorile nule</b>

# Funcții de grup



**DISTINCT** face ca funcția să ignore valorile duplicat.

**ALL** face ca funcția să afișeze și valorile duplicat.

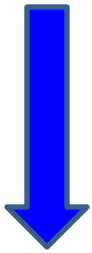
Valoarea implicită este **ALL**, deci nu este necesar să fie specificată.

Tipul de dată returnat de funcția **expr** poate fi **CHAR**, **VARCHAR2**, **NUMBER** sau **DATE**.

**Toate funcțiile de grup ignoră valorile nule.**

Pentru a lua în considerare și valorile nule se folosesc funcțiile **NVL**, **NVL2** sau **COALESCE**.





# Funcții de grup

Sintaxa funcțiilor de grup:

```
SELECT [coloana,] functie_de_grup(coloana),  
...  
FROM tabel  
[WHERE conditie]  
[GROUP BY coloana]  
[HAVING conditie_de_grupare]  
[ORDER BY coloana];
```

*Rezultatele sunt sortate **implicit crescător**.* Pentru o ordonare descrescătoare se va folosi clauza **DESC** după **ORDER BY**.

# Tabele Departament si Angajati

Pentru exemplele din cursuri vom folosi tabelele *Angajati* si *Departamente*.

## DEPARTMENT

Id\_dept int(3) CHEIE PRIMARA (PK)

Den\_dept VARCHAR (20)

Id\_manager VARCHAR (3)

Locatie VARCHAR (50)

# Tabele Angajati si Departament

```
Create Table Departament(Id_dept  
int(3) not null auto_increment  
primary key,  
Den_dept VARCHAR(20),  
Id_manager VARCHAR(3),  
Locatie VARCHAR(50)  
);
```

```
insert into  
Departament(Id_dept,Den_dept,Id_manag  
er,Locatie) VALUES  
(1,'IT',1,'Chisinau'),  
(2,'Exploatare',2,'Balti'),  
(3,'Management',1,'Chisinau'),  
(4,'HR',3,'Cahul'),  
(5,'Contabilitate',2,'Chisinau'),  
(6,'Software',4,'Anenii Noi');
```

Id_dept	Den_dept	Id_manager	Locatie
1	IT	1	Chisinau
2	Exploatare	2	Balti
3	Management	1	Chisinau
4	HR	3	Cahul
5	Contabilitate	2	Chisinau
6	Software	4	Anenii Noi

# Tabele Angajati si Departamente

## ANGAJATI

Id\_angajat **int(3)** CHEIE PRIMARA (**PK**) Id\_dept  
**int(3)** REFERINTA (**FK**) LA TABELA

## DEPARTAMENTE

Nume **VARCHAR (40)**

Prenume **VARCHAR (40)**

Functie **VARCHAR (25)**

Salariu **real(9,2)**

Id\_manager **VARCHAR (3)**

Data\_ang **DATE**

Comision **int(5)**



# Tabele Angajati si Departamente

```
create table ANGAJATI (
  Id_angajat int(3) not null auto_increment
  primary key,
  Id_dept int(3) not null,
  Nume VARCHAR(40)not null,
  Prenume VARCHAR(40)not null,
  Functie VARCHAR(25)not null,
  Salariu real(9,2)not null,
  Id_manager VARCHAR(3)not null,
  Data_ang DATE not null,
  Comision int(5)not null,
  FOREIGN KEY (Id_dept) REFERENCES
  Departament(Id_dept)
);
```

```
INSERT INTO `angajati`(`Id_angajat`, `Id_dept`,
`Nume`, `Prenume`, `Functie`, `Salariu`,
`Id_manager`, `Data_ang`, `Comision`) VALUES
('1','1','Petrov','Petru','Tester','333,20','1','22-
11-22','1'),
('2','1','Ivanov','Ion','Programator','1333,22','2',
'15-10-22','22'),
('3','2','Volodea','Petru','Designher','1500,32','
2','10-09-22','13'),
('4','3','Rebricov','Ghena','Tester','400,42','3','0
5-01-22','5),
('5','4','Vasiliev','Ion','FrontEnd
Spet','900,52','4','01-02-22','8);
```

Id_angajat	Id_dept	Nume	Prenume	Functie	Salariu	Id_manager	Data_ang	Comision
1	1	Petrov	Petru	Tester	333.00	1	2022-11-22	1
2	1	Ivanov	Ion	Programator	1333.00	2	2015-10-22	22
3	2	Volodea	Petru	Designher	1500.00	2	2010-09-22	13
4	3	Rebricov	Ghena	Tester	400.00	3	2005-01-22	5
5	4	Vasiliev	Ion	FrontEnd Spet	900.00	4	2001-02-22	8

# Funcții de grup



## Exemplul 1:

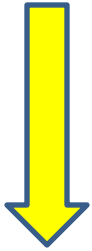
Id_angajat	Id_dept	Nume	Prenume	Funcție	Salariu	Id_ma
1	1	Petrov	Petru	Tester	333.00	1
2	1	Ivanov	Ion	Programator	1333.00	2
3	2	Volodea	Petru	Designher	1500.00	2
4	3	Rebricov	Ghena	Tester	400.00	3
5	4	Vasiliev	Ion	FrontEnd Spet	900.00	4

Afișarea salariului mediu, maxim, minim și suma tuturor salariilor angajaților cu funcție "**Tester**".

```
SELECT AVG(salariu), MAX(salariu), MIN(salariu),  
       SUM(salariu)  
FROM angajati  
WHERE functie='Tester';
```

+ Options			
AVG(salariu)	MAX(salariu)	MIN(salariu)	SUM(salariu)
366.500000	400.00	333.00	733.00

# Funcții de grup



## Exemplul 2

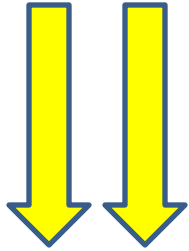
Id_angajat	Id_dept	Nume	Prenume	Funcție	Salariu	Id_manager
1	1	Petrov	Petru	Tester	333.00	1
2	1	Ivanov	Ion	Programator	1333.00	2
3	2	Volodea	Petru	Designher	1500.00	2
4	3	Rebricov	Ghena	Tester	400.00	3
5	4	Vasiliev	Ion	FrontEnd Spet	900.00	4

Datele la care s-au făcut prima și ultima angajare.

```
SELECT MIN(data_ang), MAX(data_ang)
FROM angajati;
```

+ Options	
MIN(data_ang)	MAX(data_ang)
2001-02-22	2022-11-22

# Funcții de grup



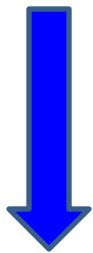
## Exemplul 3

Id_angajat	Id_dept	Nume	Prenume	Funcție	Salariu	Id_manager
1	1	Petrov	Petru	Tester	333.00	1
2	1	Ivanov	Ion	Programator	1333.00	2
3	2	Volodea	Petru	Designher	1500.00	2
4	3	Rebricov	Ghena	Tester	400.00	3
5	4	Vasiliev	Ion	FrontEnd Spet	900.00	4

Primul și ultimul nume de angajat în ordine alfabetică.

```
SELECT MIN(ume), MAX(ume)
FROM angajati;
```

+ Options	
MIN(ume)	MAX(ume)
Ivanov	Volodea



# Funcții de grup

## Funcția COUNT

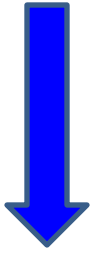
Funcția **COUNT** are 3 formate:

**COUNT(\*)**

**COUNT(expr)**

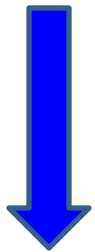
**COUNT(DISTINCT expr)**

<u>Funcția</u>	<u>Descriere</u>
<b>COUNT({* [DISTINCT ALL] expr}))</b>	Numarul de randuri unde <b>expr</b> evalueaza <b>altceva in afara de</b> <b>null</b> ( <b>folosind *</b> sunt numarate toate randurile, <b>incluzand</b> <b>duplicatale si pe cele cu</b> <b>valoare nula</b> )



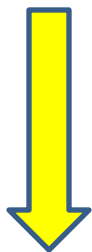
## Funcții de grup

- **COUNT(\*)** *întoarce numărul de rânduri dintr-o tabela care satisface criteriul de selecție, **incluzând** rândurile duplicat și rândurile conținând valori nule.*
- Dacă clauza **WHERE** este introdusă, atunci **COUNT(\*)** returnează numărul de rânduri care satisfac condiția din clauza WHERE.



## Funcții de grup

- În contrast, funcția **COUNT(expr)** întoarce **numărul de valori nenule** din coloana specificată de **expr**.
- **COUNT(DISTINCT expr)** returnează **numărul de valori distincte, nenule** din coloana specificată de **expr**.



# Funcții de grup

## Exemplul 4

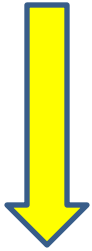
Numărul angajaților din departamentul cu id-ul '1'.

```
SELECT COUNT(*)  
FROM angajati  
WHERE id_dept = 1;
```

A screenshot of a SQL query result. It shows a single row with the text 'COUNT(\*)' in a grey box and the value '2' below it.

Id_angajat	Id_dept	Nume	Prenume	Funcție	Salariu	Id_manager	Data_ang	Comision
1	1	Petrov	Petru	Tester	333.00	1	2022-11-22	1
2	1	Ivanov	Ion	Programator	1333.00	2	2015-10-22	22
3	2	Volodea	Petru	Designher	1500.00	2	2010-09-22	13
4	3	Rebricov	Ghena	Tester	400.00	3	2005-01-22	5
5	4	Vasiliev	Ion	FrontEnd Spet	900.00	4	2001-02-22	8





# Funcții de grup

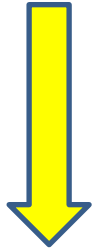
## Exemplul 5

Numărul angajaților care iau comision din departamentul '1';.

```
SELECT COUNT(comision)  
FROM angajati  
WHERE id_dept = 1
```

---

# Funcții de grup



## Exemplul 6

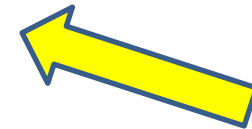
Id_angajat	Id_dept	Nume	Prenume	Funcție	Salariu	Id_manager
1	1	Petrov	Petru	Tester	333.00	1
2	1	Ivanov	Ion	Programator	1333.00	2
3	2	Volodea	Petru	Designher	1500.00	2
4	3	Rebricov	Ghena	Tester	400.00	3
5	4	Vasiliev	Ion	FrontEnd Spet	900.00	4

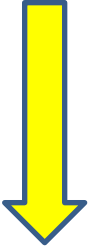
Numărul de departamente din firma  
(**varianta incorectă** și **varianta corectă**).

**SELECT** **COUNT**(id\_dept), **COUNT**(**DISTINCT** id\_dept)  
**FROM** angajati;

+ Options
COUNT(id_dept)
5

+ Options
COUNT(DISTINCT id_dept)
4





# Funcții de grup

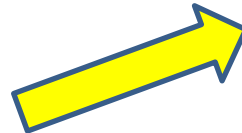
## Exemplul 7

Comisionul mediu în departamentul '1' (ignorând sau nu valorile nule).

```
SELECT AVG(comision), AVG(NVL(comision, 0))  
FROM angajati  
WHERE id_dept = 1;
```

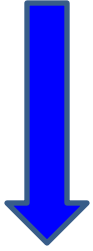
```
SELECT AVG(comision),  
AVG(COALESCE(comision, 0))  
FROM angajati  
WHERE id_dept = 1;
```

```
SELECT AVG(comision),  
AVG(IF(ISNULL(comision), 0, comision))  
FROM angajati  
WHERE id_dept = 1;
```



AVG(comision)	AVG(NVL(comision, 0))
7.0000	7.0000

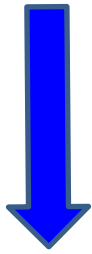
```
IF( condition, [value_if_true],  
[value_if_false] )
```



# Limbajul SQL

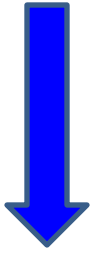
## Cereri SELECT pe o tabela

1. Funcții
2. Funcții referitoare la o singură înregistrare
3. Funcții referitoare la mai multe înregistrări
  1. Clauza **GROUP BY**
  2. Excluderea grupurilor (clauza **HAVING**)
  3. Imbricarea funcțiilor de grup




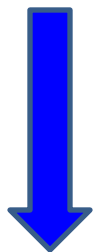
# Clauza GROUP BY

- Până acum toate funcțiile de grup au fost aplicate *întregii tabele*.
- Pentru **a putea împărți tabela în grupuri** mai mici se folosește clauza **GROUP BY**.
- Folosirea acesteia returnează *informații sumare despre fiecare grup*.

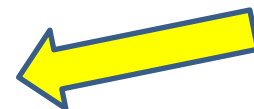


# Clauza GROUP BY

- Folosind **GROUP BY** *nu se pot extrage și coloane individuale*, ci doar *coloane ce rămân identice în tot grupul*.
- Folosind **WHERE** *se pot exclude rânduri, înaintea împărțirii lor în grupuri*. 
- *Nu pot fi folosite* aliasuri de coloane în clauza **GROUP BY**.
- Implicit, rândurile sunt sortate crescător după coloana (coloanele) specificate în **GROUP BY**.
- Acest lucru poate fi schimbat folosind **ORDER BY**.



# Clauza GROUP BY



## Exemplul 8

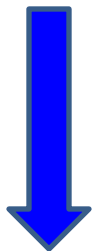
Salariul mediu pe fiecare departament.

```
SELECT id_dept,  
AVG(salariu)  
FROM angajati  
GROUP BY id_dept;
```

	id_dept	AVG(salariu)
e	1	833.000000
e	2	1500.000000
e	3	400.000000
e	4	900.000000

Id_angajat	Id_dept	Nume	Prenume	Functie	Salariu	Id_n
1	1	Petrov	Petru	Tester	333.00	1
2	1	Ivanov	Ion	Programator	1333.00	2
3	2	Volodea	Petru	Designher	1500.00	2
4	3	Rebricov	Ghena	Tester	400.00	3
5	4	Vasiliev	Ion	FrontEnd Spet	900.00	4

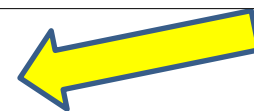
# Clauza GROUP BY



## Exemplul 9

Id_angajat	Id_dept	Nume	Prenume	Funcție	Salariu	Id_ma
1	1	Petrov	Petru	Tester	333.00	1
2	1	Ivanov	Ion	Programator	1333.00	2
3	2	Volodea	Petru	Designher	1500.00	2
4	3	Rebricov	Ghena	Tester	400.00	3
5	4	Vasiliev	Ion	FrontEnd Spet	900.00	4

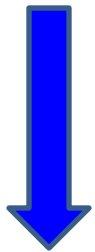
Salariul mediu pe fiecare departament, iar rezultatele ordonate după salariul mediu pe departament.



```
SELECT id_dept, AVG(salariu)
FROM angajati
GROUP BY id_dept
ORDER BY AVG(salariu);
```

dept	AVG(salariu) ▲ 1
3	400.000000
1	833.000000
4	900.000000
2	1500.000000



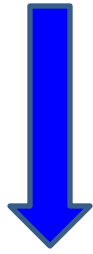


# Clauza GROUP BY

*Gruparea după mai multe coloane.*

Câteodată este necesară obținerea de rezultate pentru grupuri în alte grupuri.

Atunci în dreptul clauzei **GROUP BY** vom întâlni mai multe coloane.




# Clauza GROUP BY

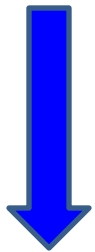
## Exemplul 10

Salariul total pe fiecare departament si pe fiecare functie, iar rezultatele ordonate după salariul mediu pe departament.

```
SELECT id_dept, functie, SUM(sa  
FROM angajati  
GROUP BY id_dept, functie  
ORDER BY AVG(salariu);
```



id_dept	functie	SUM(salariu)
1	Tester	333.00
3	Tester	400.00
4	FrontEnd Spet	900.00
1	Programator	1333.00
2	Designher	1500.00



# Limbaajul SQL

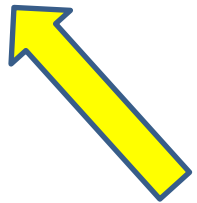
## Cereri SELECT pe o tabela

1. Funcții
2. Funcții referitoare la o singură înregistrare
3. Funcții referitoare la mai multe înregistrări
  1. Clauza **GROUP BY**
  2. Excluderea grupurilor (clauza **HAVING**)
  3. Imbricarea funcțiilor de grup



## Excluderea grupurilor (clauza **HAVING**)

Clauza **HAVING** funcționează în MARE PARTE ca și clauza **WHERE**, diferența fiind că **HAVING** *este folosit pentru a exclude anumite grupuri din rezultat*, nu rânduri cum făcea **WHERE**.



Clauza **HAVING** poate fi folosită înainte de **GROUP BY**, însă este mai logic să fie folosită după.

Ordinea execuției va rămâne aceeași.

# Excluderea grupurilor (clauza HAVING)

## Exemplul 11

Salariul mediu pe fiecare departament unde acesta depășește 3000\$.

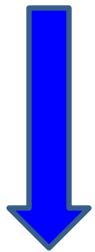
**Eroare!!**

**SELECT** id\_dept, **AVG**(salariu)  
**FROM** angajati  
**HAVING** **AVG**(salariu) > 500  
**GROUP BY** id\_dept;

SELECT id\_dept, AVG(salariu)  
FROM angajati  
GROUP BY id\_dept  
HAVING AVG(salariu) > 500

Id_angajat	Id_dept	Nume	Prenume
1	1	Petrov	Petru
2	1	Ivanov	Ion
3	2	Volodea	Petru
4	3	Rebricov	Ghena
5	4	Vasiliev	Ion

id_dept	AVG(salariu)
1	833.000000
2	1500.000000
4	900.000000

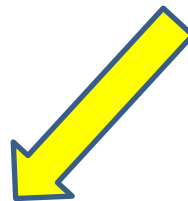


# Excluderea grupurilor (clauza HAVING)

## Exemplul 12

Salariul maxim pe fiecare departament unde acesta depășește 500\$.

```
SELECT id_dept, MAX(salariu)
FROM angajati
HAVING MAX(salariu) > 500
GROUP BY id_dept;
```



```
$sql = "SELECT id_dept,
FROM angajati\n
. "GROUP BY id_dept\n
. "HAVING MAX(salariu)"
```

id_dept	MAX(salariu)
1	1333.00
2	1500.00
4	900.00

# Excluderea grupurilor (clauza HAVING)

## Exemplul 13

Salariul total pe fiecare *funcție*, fără a lua în calcul *Testerii*, excluzând funcțiile cu suma salariilor sub 1500\$ cu ordonare după total.

Id_angajat	Id_dept	Nume	Prenume
1	1	Petrov	Petru
2	1	Ivanov	Ion
3	2	Volodea	Petru
4	3	Rebricov	Ghena
5	4	Vasiliev	Ion

**SELECT** functie, **SUM**(salariu)  
**FROM** angajati  
**WHERE** functie!='Tester'  
**GROUP BY** functie  
**HAVING** **SUM**(salariu) < 1500  
**ORDER BY** **SUM**(salariu);

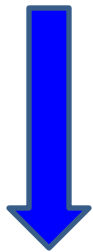
Salariul total pe fiecare funcție

fără a lua în calcul **MANAGERII**

excluzând funcțiile cu suma salariilor sub 1500\$

cu ordonare după total

functie	SUM(salariu)
FrontEnd Spet	900.00
Programator	1333.00

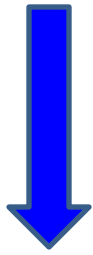


# Limbaajul SQL

## Cereri SELECT pe o tabela

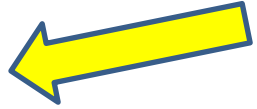
1. Funcții
2. Funcții referitoare la o singură înregistrare
3. Funcții referitoare la mai multe înregistrări
  1. Clauza **GROUP BY**
  2. Excluderea grupurilor (clauza **HAVING**)
  3. **Imbricarea funcțiilor de grup**



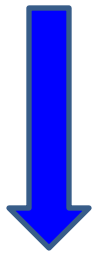


# Ordinea de executie a functiilor de grup

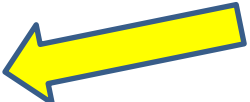


Serverul **Oracle/MYSQL** execută funcțiile de grup într-o anumită ordine:

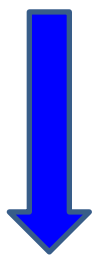


1. Selecția rândurilor ce respectă clauza **WHERE**
2. Gruparea rândurilor obținute, respectând clauza **GROUP BY**
3. Calcularea rezultatelor funcțiilor de grup pentru fiecare grup în parte
4. Eliminarea grupurilor ce nu respectă clauza **HAVING**
5. Ordonarea rezultatelor respectând clauza **GROUP BY**.



# Ordinea de executie a functiilor de grup

- *Ordinea de execuție are o importanță foarte mare, deoarece are un impact direct asupra vitezei.* 
- *Cu cât mai multe înregistrări pot fi eliminate utilizând clauza **WHERE**, cu atât mai puțin va dura gruparea și operațiile ce urmează.* 
- *Dacă o cerere **SQL** este concepută să elimine înregistrări/grupuri doar folosind clauza **HAVING**, atunci ar fi bine de încercat dacă este posibil și prin clauza **WHERE**. **De obicei, totuși, această rescriere nu va fi posibilă.*** 



# Imbricarea functiilor de grup

Funcțiile de grup pot fi imbricate cu o adâncime de 2.

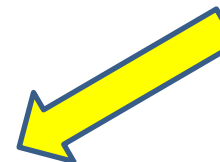
## Exemplul 14

Salariul mediu maxim.

```
SELECT  
id_dept, MAX(AVG(salariu))  
FROM angajati  
GROUP BY id_dept;
```

Стандарт также запрещает использовать агрегатную функцию как аргумент другой агрегатной функции. Т.е. мы не можем решить нашу задачу следующим образом:

Но не бывает правил без исключений. Как ни странно, но в **Oracle** подобные конструкции работают, и вышеприведенный запрос даст результат:

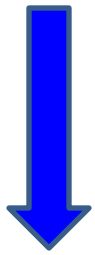


```
SELECT max(sum(salariu))  
OVER()  
FROM angajati  
GROUP BY Id_dept
```

```
SELECT DISTINCT  
max(sum(salariu)) OVER()  
FROM angajati  
GROUP BY Id_dept
```

Предложение **OVER()** не содержит дополнительного предложения **ORDER BY**, поскольку значение агрегата не зависит от сортировки строк в «окне».

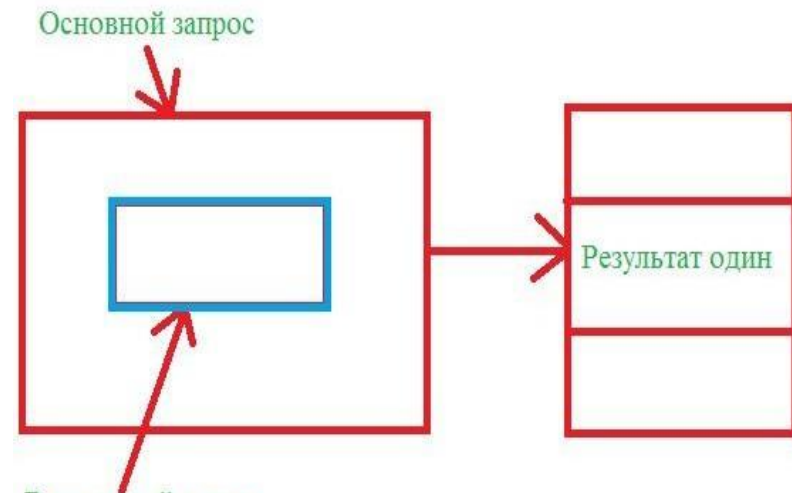
**OVER()** определяет «окно», т.е. набор строк, характеризуемых равенством значений списка выражений, указанного в этом предложении. Если предложение отсутствует, то агрегатные функции применяются ко всему результирующему набору строк запроса. В отличие от классической группировки, где мы получаем на каждую группу одну строку, которая может содержать агрегатные значения, подсчитанные для каждой такой группы, здесь мы можем добавить агрегат к детализированным (несгруппированным) строкам.



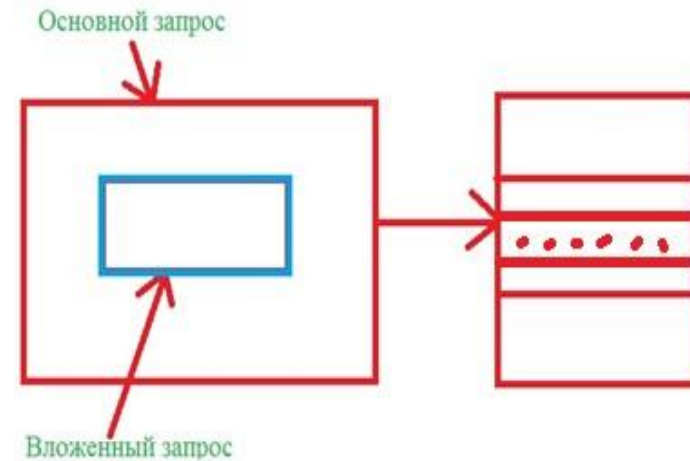
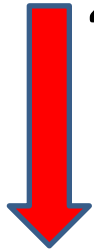
# Limbaajul SQL

## 5. Subinterogări (Subqueries)

### 1. SINGLE ROW SUBQUERIES



### 2. MULTIPLE ROW SUBQUERIES

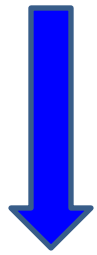


EN

<https://www.w3resource.com/sql/subqueries/understanding-sql-subqueries.php>

RU

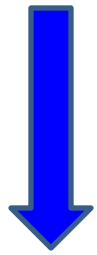
<https://www.internet-technologies.ru/articles/podzaprosy-sql.html>



# SUBQUERIES (Subinterogari)

În **SQL**, subinterogările ne permit să aflăm o informație care ne este necesară pentru a obține informația pe care o dorim s-o obținem.

- O **subinterogare (subquery)** este o instrucțiune **SELECT** care este inclusă în clauza unei alte instrucțiuni **SELECT**.



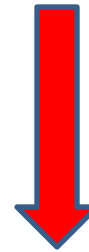
# SUBQUERIES (Subinterogari)

➤ Subinterogarea poate fi plasata în una din următoarele clauze:

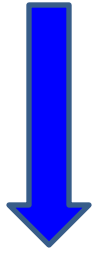
➤ WHERE

➤ HAVING

➤ FROM



➤ **Subinterogarea** se **execută prima dată**, iar rezultatul este folosit pentru obținerea rezultatului de către interogarea principală (**outer query**).

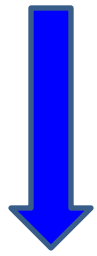


# SUBQUERIES (Subinterogari)

**Sintaxa generală:**

```
SELECT select_list  
FROM table  
WHERE expression operator  
                (SELECT select_list  
                FROM table);
```

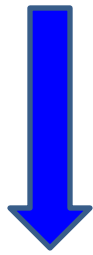




# SUBQUERIES (Subinterogari)

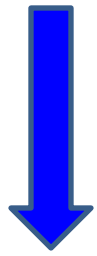
## Reguli de folosire a subinterogarilor

- O **subinterogare** se pune între paranteze rotunde
- O **subinterogare** este plasata în partea dreaptă a unei condiții de comparare
- Interogarea exterioară și **subinterogarea** pot prelua date din tabele diferite



# SUBQUERIES (Subinterogari)

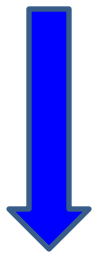
- Într-o instrucțiune **SELECT** se poate folosi o singură clauză **ORDER BY** și, dacă se folosește, trebuie să fie ultima clauza a interogării principale.
- Un subquery **nu poate avea** propria clauză **ORDER BY**.
- Singura limită a numărului de interogări este dimensiunea buffer-ului folosit de interogare.
- Dacă subinterogarea returnează **null** sau nu returnează nici o linie, atunci interogarea exterioară nu va returna nimic.



# SUBQUERIES (Subinterogari)

Sunt două tipuri de subinterogări(subqueries):

- 1) **single-row subqueries** – care folosesc operatorii single-row: **>, =, >=, <, <=** și dau ca rezultat o **singură linie**.
- 2) **multiple-row subqueries** – care folosesc operatorii multiple-row: **IN, ANY, ALL** și dau ca rezultat **mai multe linii**.



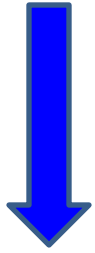
# Limbaajul SQL

## 5. SUBQUERIES (Subinterogări)

### 1. SINGLE ROW SUBQUERIES

### 2. MULTIPLE ROW SUBQUERIES

Id_angajat	Id_dept	Nume	Prenume	Funcție	Salariu	Id_manager	Data_ang	Comision
1	1	Petrov	Petru	Tester	333.00	1	2022-11-22	1
2	1	Ivanov	Ion	Programator	1333.00	2	2015-10-22	22
3	2	Volodea	Petru	Designher	1500.00	2	2010-09-22	13
4	3	Rebricov	Ghena	Tester	400.00	3	2005-01-22	5
5	4	Vasiliev	Ion	FrontEnd Spet	900.00	4	2001-02-22	8



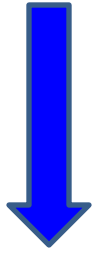
# SINGLE ROW SUBQUERIES

## Single row-subquery

```
SELECT nume, salariu  
FROM angajati  
WHERE Id_angajat =
```

```
( SELECT Id_angajat  
  FROM angajati  
  WHERE Prenume= 'Ghena' );
```

Aflati prenumele angajatilor  
care au salariul mai mare  
decat angajatul care se  
numeste Ghena.



# Multiple ROW SUBQUERIES

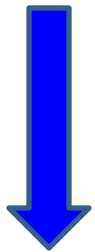
## Multiple row-subquery

```
SELECT prenume  
FROM angajati  
WHERE salariu > Any [IN]
```

Aflati prenumele angajatilor  
care au salariul mai mare  
decat angajatul care se  
numeste Petru.

```
( SELECT salariu  
FROM angajati  
WHERE prenume='Petru' );
```

```
SELECT AUTOR  
FROM carti  
WHERE PRET >  
( SELECT AVG(PRET)
```



# SINGLE ROW SUBQUERIES

## Subcereri din mai multe tabele

**Subcererile (subinterogările) nu sunt limitate la o singură interogare (cerere).**

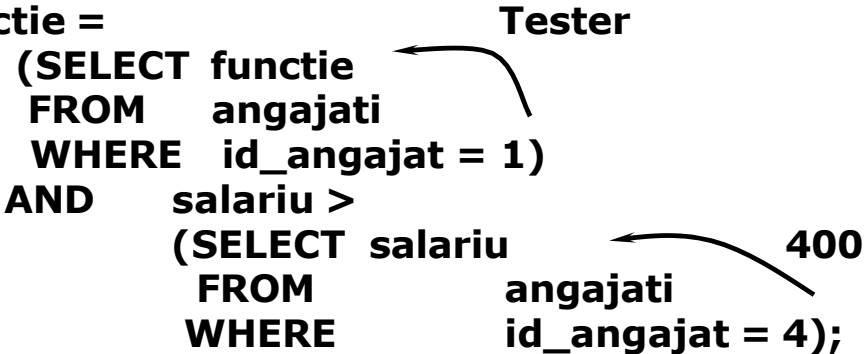
Așa cum se poate observa în exemplul următor, pot fi mai mult de o singură interogare.

De asemenea se pot face interogări din tabele diferite.

Exemplul urmator afiseaza angajatii a caror functie este aceiasi cu cei al angajatului cu numarul 7369 si a caror salariu este mai mare decat cel al angajatului 7875.

## Executarea unei subinterogari single-row

```
SELECT  nume, functie
FROM    angajati
WHERE   functie =
        (SELECT functie
         FROM    angajati
         WHERE   id_angajat = 1)
AND     salariu >
        (SELECT salariu
         FROM    angajati
         WHERE   id_angajat = 4);
```



nume	Funcție
Petrov	Tester



# SINGLE ROW SUBQUERIES

- Exemplul este format din 3 blocuri de cereri:
  - o cerere exterioara
  - doua cereri interne
- Blocurile de cereri interne sunt primele executate, producand rezultatele cererii: **Tester** respectiv **400**
- Blocul exterior de cereri este apoi procesat si foloseste valorile returnate de catre cererile interne pentru a finaliza propriile conditii de cautare.
- Ambele cereri interne returneaza valori singulare (**Tester si 400**), astfel ca aceasta instructiune SQL este denumita o subinterogare single-row.

# SINGLE-ROW SUBQUERIES

**SELECT** nume, functie, salariu, id\_dept  
**FROM** angajati  
**WHERE** functie =

(**SELECT** functie  
**FROM** angajati  
**WHERE** id\_angajat=5)

**AND** id\_dept =

(**SELECT** id\_dept  
**FROM** departamente  
**WHERE** locatie='Cahul');

Id_dept	Den_dept	Id_n
1	IT	1
2	Exploatare	2
3	Management	1
4	HR	3
5	Contabilitate	2
6	Software	4

Funcție	Salariu	Id_manager	Data_ang	Comision
Tester	333.00	1	2022-11-22	1
Programator	1333.00	2	2015-10-22	22
Designher	1500.00	2	2010-09-22	13
Tester	400.00	3	2005-01-22	5
FrontEnd	900.00	4	2001-02-22	8

# SINGLE ROW SUBQUERIES

Se pot folosi funcțiile de grup în subinterogări.

O funcție de grup utilizată în subquery fără clauza  
**GROUP BY**, *returnează o singură linie*.

**SELECT** nume, prenume, salariu  
**FROM** angajati  
**WHERE** salariu <

(**SELECT** **MAX**(salariu)  
**FROM** angajati);

nume	prenume	salariu
Petrov	Petru	333.00
Ivanov	Ion	1333.00
Rebricov	Ghena	400.00
Vasiliev	Ion	900.00

# SINGLE ROW SUBQUERIES

- Subinterogările pot fi plasate și în clauza **HAVING**.
- Deoarece clauza **HAVING** *are întotdeauna o condiție de grup, și subinterogarea va avea aproape întotdeauna o condiție de grup.*

```
SELECT id_dept, MIN(salariu)
FROM angajati
GROUP BY id_dept
HAVING MIN(salariu) >
```

id_dept	MIN(salariu)
2	1500.00
3	400.00
4	900.00

```
(SELECT MIN(salariu)
FROM angajati
WHERE id_dept = 1);
```

# SINGLE ROW SUBQUERIES

1) Care este rezultatul?

**SELECT** nume  
**FROM** angajati  
**WHERE** salariu >

nume	Salariu	Id_manager	Data_ang	Comision
Angajator	333.00	1	2022-11-22	1
Angajator	1333.00	2	2015-10-22	22
Angajator	1500.00	2	2010-09-22	13
Angajator	400.00	3	2005-01-22	5
Angajator	900.00	4	2001-02-22	8

(**SELECT** salariu  
**FROM** angajati  
**WHERE** id\_angajat = 3);

Id_dept	Den_dept
1	IT
2	Exploatare
3	Management
4	HR
5	Contabilitate
6	Software

# SINGLE ROW SUBQUERIES

2) Care este rezultatul? Comentati...

**SELECT** nume, prenume  
**FROM** angajati  
**WHERE** id\_dept =

	Salariu	Id_manager	Data_ang	Comision
	333.00	1	2022-11-22	1
ator	1333.00	2	2015-10-22	22
ner	1500.00	2	2010-09-22	13
	400.00	3	2005-01-22	5
d	900.00	4	2001-02-22	8

( **SELECT** id\_dept  
**FROM** departament  
**WHERE** Den\_dept = 'IT' );

Id_dept	Den_dept	Id_r
1	IT	1
2	Exploatare	2
3	Management	1
4	HR	3
5	Contabilitate	2
6	Software	4

# Limbaajul SQL

## 5. SUBQUERIES (Subinterogări)

### 1. SINGLE ROW SUBQUERIES

### 2. MULTIPLE ROW SUBQUERIES

# MULTIPLE ROW SUBQUERIES

*Sunt acele subinterogări care dau ca rezultat mai multe valori.*

Folosesc operatorii **multiple row**:

1. **IN**
2. **ANY**
3. **ALL**

Operatorul **NOT** poate fi folosit în combinație cu oricare dintre aceștia.



# MULTIPLE ROW SUBQUERIES

## 1. Operatorul IN

Operatorul **IN** este folosit dacă în interogarea exterioară clauza **WHERE** este folosită pentru a selecta acele *valori care sunt egale cu una dintre valorile din lista returnată de subinterogare* (inner query).

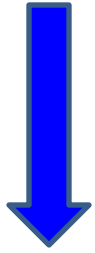
```
SELECT nume, salariu, id_dept  
FROM angajati  
WHERE salariu IN  
    ( SELECT MIN(salariu)  
      FROM angajati  
      GROUP BY id_dept );
```

nume	salariu	id_dept
Petrov	333.00	1
Volodea	1500.00	2
Rebricov	400.00	3
Vasiliev	900.00	4

# MULTIPLE ROW SUBQUERIES

## 2. Operatorul ANY

Acest operator este folosit atunci când dorim ca interogarea exterioară să selecteze valori egale, mai mici sau mai ***mari decât cel puțin o valoare dintre cele extrase de subquery.***



# Multiple ROW SUBQUERIES

## Multiple row-subquery

```
SELECT prenume  
FROM angajati  
WHERE salariu > Any [IN]
```

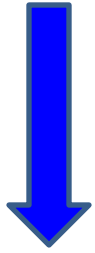
```
( SELECT salariu  
  FROM angajati  
  WHERE prenume='Petru' );
```

Aflati prenumele angajatilor  
care au salariul mai mare  
decat angajatul care se  
numeste Petru.

# MULTIPLE ROW SUBQUERIES

## 3. Operatorul ALL

Acest operator este folosit atunci când dorim ca interogarea exterioară să selecteze valori egale, mai mici sau mai mari decât toate valorile extrase de subquery.



# Multiple ROW SUBQUERIES

## Multiple row-subquery

```
SELECT prenume  
FROM angajati  
WHERE salariu > All
```

Aflati prenumele angajatilor  
care au salariul mai mare  
decat angajatul care se  
numeste Petru.

```
( SELECT salariu  
  FROM angajati  
  WHERE prenume='Petru' );
```

# MULTIPLE ROW SUBQUERIES

## VALORI NULL

Dacă una dintre valorile returnate de subinterogarea **multiple row** este **null**, dar celelalte valori nu sunt **null**, atunci:

- Dacă sunt folosiți operatorii **IN** sau **ANY**, interogarea exterioară va returna liniile care se potrivesc cu valorile **non-null**.
- Dacă este folosit operatorul **ALL**, interogarea exterioară nu va returna nimic.

# MULTIPLE ROW SUBQUERIES

Clauzele **GROUP BY** și **HAVING**

- Pot fi folosite cu subinterogările de tip **MULTIPLE ROW**.

```
SELECT id_dept, MIN(salariu)
FROM angajati
GROUP BY id_dept
HAVING MIN(salariu) < ANY
      ( SELECT salariu
        FROM angajati
        WHERE id_dept IN (1,2) );
```

id_dept	MIN(salariu)
1	333.00
3	400.00
4	900.00

# MULTIPLE ROW SUBQUERIES

## Clauzele **GROUP BY** si **HAVING**

De asemenea, se poate folosi clauza **GROUP BY** intr-o subinterogare

```
SELECT id_dept, MIN(salariu)
FROM angajati
GROUP BY id_dept
HAVING MIN(salariu) > ALL
      (SELECT MIN(salariu)
       FROM angajati
       WHERE id_dept < 2
       Group by id_dept);
```

id_dept	MIN(salariu)
2	1500.00
3	400.00
4	900.00



# MULTIPLE ROW SUBQUERIES

## APLICATII

- 1) Găsiți numele pentru toți angajații ale căror salarii sunt aceleași cu salariul minim din oricare (any) departament. Explicati rez.

```
SELECT nume
FROM angajati
WHERE salariu = ANY
      (SELECT MIN(salariu)
       FROM angajati
       GROUP BY id_dept);
```

nume

⌵ Petrov

⌵ Volodea

⌵ Rebricov

⌵ Vasiliev

# MULTIPLE ROW SUBQUERIES

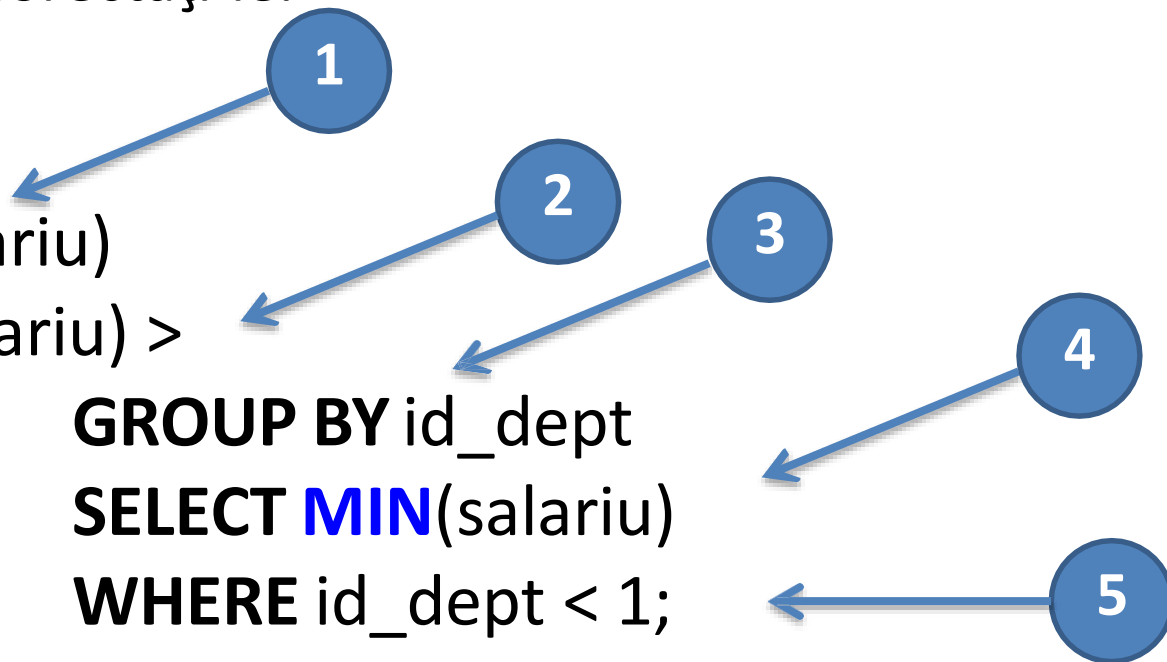
2) Scopul interogării următoare este de a afișa salariul minim pentru fiecare departament al cărui salariu minim este mai mic decât cel mai mic salariu al angajaților din departamentul 2.

Oricum, subinterogarea nu se execută deoarece are 5 erori.

Găsiți erorile și corectați-le.

```
SELECT id_dept  
FROM angajati  
WHERE MIN(salariu)  
HAVING MIN(salariu) >
```

```
GROUP BY id_dept  
SELECT MIN(salariu)  
WHERE id_dept < 1;
```



# MULTIPLE ROW SUBQUERIES

Soluția corectă este următoarea:

```
SELECT id_dept, MIN(salariu)
FROM angajati
GROUP BY id_dept
HAVING MIN(salariu) <
      (SELECT MIN(salariu)
       FROM angajati
       WHERE id_dept = 1);
```

## Subcereri multilinie

- Subcererile multilinie returneaza mai mult decat o linie.
- Cu astfel de subcereri trebuie folositi operatori multilinie care pot prelucra una sau mai multe valori.

Operatorii utilizati sunt:

1. **IN** - egal cu oricare dintre membrii unei liste
2. **ANY/SOME** - compara o valoare cu fiecare (vreo) valoare returnata de subcerere
3. **ALL** - compara o valoare cu oricare (toate) din valorile returnate de subcerere

## Exemplu IN

- Aflati angajatii care au salariul egal cu salariul cel mai mare din fiecare departament

```
SELECT nume, id_dept,  
salariu FROM angajati  
WHERE salariu IN
```

```
(SELECT  
MAX(salariu)  
FROM angajati  
GROUP BY  
id_dept)
```

nume	id_dept	salariu
Ivanov	1	1333.00
Volodea	2	1500.00
Rebricov	3	400.00
Vasiliev	4	900.00

Subcererea ofera salariile maxime din fiecare departament si prin cererea principala se afla angajatii cu aceste salarii.

## Exemplu **ANY**

Aflati angajatii care au salariul mai mare decat vreun angajat al departamentului 2 si nu fac parte din acest departament.

**SELECT nume, id\_dept, salariu**  
**FROM angajati**  
**WHERE salariu > ANY**  
**( SELECT salariu**  
**FROM angajati**  
**WHERE id\_dept=2)**  
**AND id\_dept<>2**

Id_dept	Den_dept	Id_manager
1	IT	1
2	Exploatare	2
3	Management	1
4	HR	3
5	Contabilitate	2
6	Software	4

Id_angajat	Id_dept	Nume	Prenume	Funcctie	Salariu	Id_manager	Data_ang	Comision
1	1	Petrov	Petru	Tester	333.00	1	2022-11-22	1
2	1	Ivanov	Ion	Programator	1333.00	2	2015-10-22	22
3	2	Volodea	Petru	Designher	1500.00	2	2010-09-22	13
4	3	Rebricov	Ghena	Tester	400.00	3	2005-01-22	5
5	4	Vasiliev	Ion	FrontEnd Spet	900.00	4	2001-02-22	8

- Operatorul **ANY** (sinonim operatorului **SOME**) compara o valoare cu fiecare valoare din cele returnate de subcerere.

Astfel,

**< ANY** inseamna mai mic decat maximul

**> ANY** inseamna mai mare decat minimul

**= ANY** este echivalent cu **IN**



## Exemplu ALL

Gasiti angajatii care au salariul mai mic decat oricare (toti) angajatii de la departamentul 5.

```
SELECT nume, id_dept, salariu  
FROM angajati  
WHERE salariu < ALL  
      ( SELECT salariu  
        FROM angajati  
        WHERE id_dept=5 )  
AND id_dept<>5
```

	nume	id_dept	salariu
e	Petrov	1	333.00
e	Ivanov	1	1333.00
e	Volodea	2	1500.00
e	Rebricov	3	400.00
e	Vasiliev	4	900.00

- Operatorul **ALL** din cererea principala compara o valoare cu oricare valoare returnata de subcerere.

Astfel:

**> ALL** inseamna mai mare decat maximul

**< ALL** inseamna mai mic decat minimul

# Imbricarea subcererilor

Subcererile pot fi folosite si in interiorul altor subcereri.

## Exemplu

Gasiti numele, functia, data angajarii si salariul angajatilor al caror salariu este superior celui mai mare salariu al vreunei persoane angajate dupa data de 2015- 10-22

Id_angajat	Id_dept	Nume	Prenume	Functie	Salariu	Id_manager	Data_ang
1	1	Petrov	Petru	Tester	333.00	1	2022-11-22
2	1	Ivanov	Ion	Programator	1333.00	2	2015-10-22
3	2	Volodea	Petru	Designher	1500.00	2	2010-09-22
4	3	Rebricov	Ghena	Tester	400.00	3	2005-01-22
5	4	Vasiliev	Ion	FrontEnd Spet	900.00	4	2001-02-22

```

SELECT nume, functie, data_ang, salariu
      FROM angajati
     WHERE salariu >
(SELECT MAX(salariu)
 FROM angajati
 WHERE data_ang IN
      (SELECT data_ang
        FROM angajati
       WHERE data_ang > '22-10-2015'))

```

	nume	functie	data_ang	salariu
e	Ivanov	Programator	2015-10-22	1333.00
e	Volodea	Designher	2010-09-22	1500.00
e	Rebricov	Tester	2005-01-22	400.00
3.2.4 [ Compiled: Jun 5th 2019 ]			01-02-22	900.00

Numarul maxim de imbricari pentru o subcerere este de 255.

## SUBCERERI CORELATE

- O subcerere corelata este o subcerere care se executa o data pentru fiecare linie considerata de cererea principala si care la executie foloseste o valoare dintr-o coloana din cererea exterioara.
- Ea se poate identifica prin folosirea unei coloane a cererii exterioare in clauza operatorului cererii interioare.

## Exemplu

Gasiti angajatii care au un salariu superior salariului mediu al departamentului lor. \_\_\_\_\_

```
SELECT nume, salariu, id_dept  
FROM angajati A  
WHERE salariu >
```

```
    (SELECT AVG(salariu)  
     FROM angajati  
     WHERE (id_dept=A.id_dept))
```

```
ORDER BY id_dept
```

salariu	id_dept
1333.00	1

## Valori de NULL intr-o subcerere

- In cazul in care subcererea returneaza vreuna din valori NULL si cererea principala are operator NOT IN, atunci cererea principala nu va returna niciun rand.
- Motivul este ca *o comparatie cu NULL conduce la un rezultat NULL*.

Chisinau
Balti
Chisinau
Cahul
Chisinau
Anenii Noi

## Exemplu

Id_angajat	Id_dept	Nume	Prenume	Funcție	Salariu	Id_manager
1	1	Petrov	Petru	Tester	333.00	1
2	1	Ivanov	Ion	Programator	1333.00	2
3	2	Volodea	Petru	Designher	1500.00	2
4	3	Rebricov	Ghena	Tester	400.00	3
5	4	Vasiliev	Ion	FrontEnd Spet	900.00	4

Gasiti angajatii care nu au subordonati.

```
SELECT nume FROM
angajati
WHERE id_angajat NOT IN
( SELECT id_manager
FROM angajati );
```

Options

T→

nume

] Edit Copy Delete Vasiliev

☐ Check all With selected:

☐ Show all | Number of rows: 25



- Astfel ori de cate ori valoarea **NULL** face parte din raspunsurile subcererii nu trebuie folosit operatorul **NOT IN**.
- De fapt operatorul **NOT IN** este echivalent cu **<> ALL**.
- Returnarea de valori NULL de catre subcerere nu prezinta nici o problema in cazul operatorului **IN** in cererea principala (in echivalent cu **= ALL**).

## Exemplu

Gasiti angajatii care au subordonati.

```
SELECT nume  
FROM angajati  
WHERE id_angajat IN  
    ( SELECT id_manager  
      FROM angajati );
```

n_dept	Id_manager	Locatie
	1	Chisinau
exploatare	2	Balti
management	1	Chisinau
	3	Cahul
stabilitate	2	Chisinau

Id_angajat	Id_dept	Nume	Prenume	Functie
1	1	Petrov	Petru	Test
2	1	Ivanov	Ion	Prog
3	2	Volodea	Petru	Des
4	3	Rebricov	Ghena	Test
5	4	Vasiliev	Ion	Front Spe

nume
e Petrov
e Ivanov
e Volodea
e Rebricov

selected:

- In cazul utilizarii operatorului **NOT IN** in cererea principala trebuie avut grija sa se excluda valorile **NULL** din raspunsurile subcererii.

## **Exemplu**

Gasiti angajatii care nu au subordonati.

```

SELECT nume FROM
angajati
WHERE id_angajat NOT IN
      ( SELECT id_manager FROM
angajati
WHERE id_manager IS NOT NULL );

```

dept	Den_dept	Id_manager	Locatie
1	IT	1	Chisinau
2	Exploatare	2	Balti
3	Management	1	Chisinau
4	HR	3	Cahul
5	Contabilitate	2	Chisinau
6	Software	4	Anenii Noi

Id_angajat	Id_dept	Nume	Prenume	Funcție	Sala
1	1	Petrov	Petru	Tester	333
2	1	Ivanov	Ion	Programator	1333
3	2	Volodea	Petru	Designher	1500
4	3	Rebricov	Ghena	Tester	400
5	4	Vasiliev	Ion	FrontEnd Spet	900

## Sfaturi în utilizarea subinterogărilor

1. Includerea subinterogărilor în paranteze
2. Plasarea subinterogărilor în partea dreapta a operatorului de comparare
3. A nu se adauga clauza **ORDER BY** într-o subinterogare
4. Folosirea operatorilor single-row în subinterogari single-row
5. Folosirea operatorilor multiple-row în subinterogari multiple-row

# Concluzii

1. O subinterogare este o instructiune **SELECT** inclusa într-o clauza a altei instructiuni **SQL**.
2. Subinterogările sunt folosite atunci când interogarea se bazează pe criterii necunoscute.
3. Subinterogările au următoarele caracteristici:
  - a) Pot transmite un rand de date instructiunii principale care contine un **operator single-row**, precum: **=, <>, >, >=, <** sau **<=**;
  - b) Pot transmite rînduri multiple de date instructiunii principale care contine un **operator multiple-row**, precum: **IN, ANY** sau **ALL**;
  - c) Sunt primele procesate de către server-ul **Oracle**, iar clauzele **WHERE** si **HAVING** folosesc rezultatele;
  - d) Pot contine functii de grup

