

Funcții SQL. Cereri multi-relație (introducere)

I. [Funcții SQL]

Funcțiile SQL sunt predefinite în sistemul *Oracle* și pot fi utilizate în instrucțiuni SQL. Ele nu trebuie confundate cu funcțiile definite de utilizator, scrise în *PL/SQL*.

Dacă o funcție SQL este apelată cu un argument având un alt tip de date decât cel așteptat, sistemul convertește implicit argumentul înainte să evalueze funcția.

De obicei, dacă o funcție SQL este apelată cu un argument *null*, ea returnează valoarea *null*. Funcțiile care nu urmează această regulă sunt *CONCAT*, *NVL* și *REPLACE*.

Funcțiile SQL pot fi clasificate în următoarele categorii:

- Funcții *single-row*
- Funcții *multiple-row* (funcții agregat)

1. Funcțiile *single row* returnează câte o singură linie rezultat pentru fiecare linie a tabelului sau vizualizării interogate. Aceste funcții pot apărea în:

- listele de expresii din clauza *SELECT*
- clauzele *WHERE*, *START WITH*, *CONNECT BY* și *HAVING*.

În ceea ce privește tipul argumentelor asupra cărora operează și al rezultatelor furnizate, funcțiile *single row* pot fi clasificate în categorii corespunzătoare.

❑ **Funcțiile de conversie** cele mai importante sunt:

Funcție	Descriere	Exemplu conversie
<i>TO_CHAR</i>	convertește (sau formatează) un număr sau o dată calendaristică în șir de caractere	<i>TO_CHAR</i> (7) = '7' <i>TO_CHAR</i> (-7) = '-7' <i>TO_CHAR</i> (SYSDATE, 'DD/MM/YYYY') = '02/03/2016'
<i>TO_DATE</i>	convertește (sau formatează) un număr sau un șir de caractere în dată calendaristică	<i>TO_DATE</i> ('18-MAR-2016','dd-mon-yyyy')
<i>TO_NUMBER</i>	convertește (sau formatează) un șir de caractere în număr	<i>TO_NUMBER</i> (' -25789', 'S99,999') = -25,789

Obs: Există două tipuri de conversii:

- **implicite**, realizate de sistem atunci când este necesar;
- **explicite**, indicate de utilizator prin intermediul funcțiilor de conversie.

Conversiile implicite asigurate de server-ul *Oracle* sunt:

- de la *VARCHAR2* sau *CHAR* la *NUMBER*;
- de la *VARCHAR2* sau *CHAR* la *DATE*;
- de la *NUMBER* la *VARCHAR2* sau *CHAR*;
- de la *DATE* la *VARCHAR2* sau *CHAR*.

□ Dintre funcțiile pentru prelucrarea caracterelor amintim:

Funcție	Descriere	Exemplu
<i>LENGTH(string)</i>	întoarce lungimea șirului de caractere <i>string</i>	<i>LENGTH('Informatica')=11</i>
<i>SUBSTR(string, start [,n])</i>	întoarce subșirul lui <i>string</i> care începe pe poziția <i>start</i> și are lungimea <i>n</i> ; dacă <i>n</i> nu este specificat, subșirul se termină la sfârșitul lui <i>string</i> ;	<i>SUBSTR('Informatica', 1, 4) = 'Info'</i> <i>SUBSTR('Informatica', 6) = 'matica'</i> <i>SUBSTR('Informatica', -5) = 'matica'</i> (ultimele 5 caractere)
<i>LTRIM(string [, 'chars'])</i>	șterge din stânga șirului <i>string</i> orice caracter care apare în <i>chars</i> , până la găsirea primului caracter care nu este în <i>chars</i> ; în cazul în care <i>chars</i> nu este specificat, se șterg spațiile libere din stânga lui <i>string</i> ;	<i>LTRIM (' info') = 'info'</i>
<i>RTRIM(string [, 'chars'])</i>	este similar funcției <i>LTRIM</i> , cu excepția faptului că ștergerea se face la dreapta șirului de caractere;	<i>RTRIM ('infoXXXX', 'X') = 'info'</i>
<i>TRIM (LEADING TRAILING BOTH chars FROM expresie)</i>	elimină caracterele specificate (<i>chars</i>) de la începutul (<i>leading</i>) , sfârșitul (<i>trailing</i>) sau din ambele părți, dintr-o expresie caracter dată.	<i>TRIM (LEADING 'X' FROM 'XXXInfoXXX') = 'InfoXXX'</i> <i>TRIM (TRAILING 'X' FROM 'XXXInfoXXX') = 'XXXInfo'</i> <i>TRIM (BOTH 'X' FROM 'XXXInfoXXX') = 'Info'</i> <i>TRIM (BOTH FROM ' Info ') = 'Info'</i>
<i>LPAD(string, length [, 'chars'])</i>	adaugă <i>chars</i> la stânga șirului de caractere <i>string</i> până când lungimea noului șir devine <i>length</i> ; în cazul în care <i>chars</i> nu este specificat, atunci se adaugă spații libere la stânga lui <i>string</i> ;	<i>LPAD (LOWER('info'),6) = ' info'</i>
<i>RPAD(string, length [, 'chars'])</i>	este similar funcției <i>LPAD</i> , dar adăugarea de caractere se face la dreapta șirului;	<i>RPAD (LOWER('info'), 6, 'X') = 'infoXX'</i>
<i>REPLACE(string1, string2 [,string3])</i>	întoarce <i>string1</i> cu toate aparițiile lui <i>string2</i> înlocuite prin <i>string3</i> ; dacă <i>string3</i> nu este specificat, atunci toate aparițiile lui <i>string2</i> sunt șterse;	<i>REPLACE ('\$b\$bb','\$','a') = 'ababb'</i> <i>REPLACE ('\$b\$bb','\$b','ad') = 'adadb'</i> <i>REPLACE ('\$a\$aa','\$') = 'aaa'</i>
<i>UPPER(string), LOWER(string)</i>	transformă toate literele șirului de caractere <i>string</i> în majuscule, respectiv minuscule;	<i>LOWER ('info') = 'info'</i> <i>UPPER ('info') = 'INFO'</i>
<i>INITCAP(string)</i>	transformă primul caracter al șirului în majusculă, restul caracterelor fiind transformate în minuscule	<i>INITCAP ('info') = 'Info'</i>

<i>INSTR(string, 'chars' [,start [,n]])</i>	caută în <i>string</i> , începând de la poziția <i>start</i> , a <i>n</i> -a apariție a secvenței <i>chars</i> și întoarce poziția respectivă; dacă <i>start</i> nu este specificat, căutarea se face de la începutul șirului; dacă <i>n</i> nu este specificat, se caută prima apariție a secvenței <i>chars</i> ;	<i>INSTR (LOWER('AbC aBcDe'), 'ab', 5, 2)</i> = 0 <i>INSTR (LOWER('AbCdE aBcDe'), 'ab', 5)</i> = 7
<i>ASCII(char)</i>	furnizează codul <i>ASCII</i> al primului caracter al unui șir	<i>ASCII ('alfa') = ASCII ('a') = 97</i>
<i>CHR(num)</i>	întoarce caracterul corespunzător codului <i>ASCII</i> specificat	<i>CHR(97)= 'a'</i>
<i>CONCAT(string1, string2)</i>	realizează concatenarea a două șiruri de caractere	<i>CONCAT ('In', 'fo') = 'Info'</i>
<i>TRANSLATE(string, source, destination)</i>	fiecare caracter care apare în șirurile de caractere <i>string</i> și <i>source</i> este transformat în caracterul corespunzător (aflat pe aceeași poziție ca și în <i>source</i>) din șirul de caractere <i>destination</i>	<i>TRANSLATE('\$a\$aa','\$','b') = 'babaa'</i> <i>TRANSLATE('\$a\$aaa','\$a','bc') = 'bcbccc'</i>

Obs: Testarea funcțiilor prezentate se face de maniera : *SELECT apel_funcție FROM dual;* astfel că vom omite comanda *SELECT* și vom da numai apelul funcției și rezultatul returnat.

❑ **Funcțiile aritmetice** *single-row* pot opera asupra:

- unei singure valori, și aceste funcții sunt: *ABS* (valoarea absolută), *CEIL* (partea întreagă superioară), *FLOOR* (partea întreagă inferioară), *ROUND* (rotunjire cu un număr specificat de zecimale), *TRUNC* (trunchiere cu un număr specificat de zecimale), *EXP* (ridicarea la putere a lui *e*), *LN* (logaritm natural), *LOG* (logaritm într-o bază specificată), *MOD* (restul împărțirii a două numere specificate), *POWER* (ridicarea la putere), *SIGN* (semnul unui număr), *COS* (cosinus), *COSH* (cosinus hiperbolic), *SIN* (sinus), *SINH* (sinus hiperbolic), *SQRT* (rădăcina pătrată), *TAN* (tangent), *TANH* (tangent hiperbolic);
- unei liste de valori, iar acestea sunt funcțiile *LEAST* și *GREATEST*, care întorc cea mai mică, respectiv cea mai mare valoare a unei liste de expresii.

❑ **Funcțiile pentru prelucrarea datelor calendaristice** sunt:

Funcție	Descriere	Exemplu
<i>SYSDATE</i>	întoarce data și timpul curent	<i>SELECT SYSDATE FROM dual;</i> (de revăzut utilizarea acestei funcții împreună cu <i>TO_CHAR</i> în cadrul laboratorului 1)
<i>ADD_MONTHS(expr_date, nr_luni)</i>	întoarce data care este după <i>nr_luni</i> luni de la data <i>expr_date</i> ;	<i>ADD_MONTHS('02-MAR-2016', 3) = '02-JUN-2016'.</i>
<i>NEXT_DAY(expr_date, day)</i>	întoarce următoarea dată după data <i>expr_date</i> , a cărei zi a săptămânii este cea specificată prin șirul de caractere <i>day</i>	<i>NEXT_DAY('02-MAR-2016', 'Monday') = '07-MAR-2007'</i>

<i>LAST_DAY(expr_date)</i>	întoarce data corespunzătoare ultimei zile a lunii din care data <i>expr_date</i> face parte	<i>LAST_DAY('02-MAR-2016') = '31-MAR-2016'</i>
<i>MONTHS_BETWEEN(expr_date2, expr_date1)</i>	întoarce numărul de luni dintre cele două date calendaristice specificate. Data cea mai recentă trebuie specificată în primul argument, altfel rezultatul este negativ.	<i>MONTHS_BETWEEN('02-DEC-2014', '10-OCT-2011') = 37.7419355</i> <i>MONTHS_BETWEEN('10-OCT-2011', '02-DEC-2014') = -37.7419355</i>
<i>TRUNC(expr_date)</i>	întoarce data <i>expr_date</i> , dar cu timpul setat la ora 12:00 AM (miezul nopții)	<i>TO_CHAR(TRUNC(SYSDATE), 'dd/mm/yy HH24:MI') = '02/03/16 00:00'</i>
<i>ROUND(expr_date)</i>	dacă data <i>expr_date</i> este înainte de miezul zilei, întoarce data <i>d</i> cu timpul setat la ora 12:00 AM; altfel, este returnată data corespunzătoare zilei următoare, cu timpul setat la ora 12:00 AM	<i>TO_CHAR(ROUND(SYSDATE), 'dd/mm/yy hh24:mi am') = '02/03/16 00:00 AM'</i>
<i>LEAST(d1, d2, ..., dn), GREATEST(d1, d2, ..., dn)</i>	dintr-o listă de date calendaristice, funcțiile întorc prima, respectiv ultima dată în ordine cronologică	<i>LEAST(SYSDATE, SYSDATE + 3, SYSDATE - 5) = SYSDATE-5</i> <i>GREATEST(SYSDATE, SYSDATE + 3, SYSDATE - 5) = SYSDATE + 3</i>

Operațiile care se pot efectua asupra datelor calendaristice sunt următoarele:

Operație	Tipul de date al rezultatului	Descriere
<i>expr_date</i> +/- <i>expr_number</i>	<i>Date</i>	Scade/adună un număr de zile dintr-o / la o dată. Numărul de zile poate să nu fie întreg (putem adăuga, de exemplu, un număr de minute sau de ore).
<i>expr_date1</i> – <i>expr_date2</i>	<i>Number</i>	Întoarce numărul de zile dintre două date calendaristice. Data <i>expr_date1</i> trebuie să fie mai recentă decât <i>expr_date2</i> , altfel rezultatul este negativ.

❑ Funcții diverse:

Funcție	Descriere	Exemplu
<i>DECODE(value, if1, then1, if2, then2, ..., ifN, thenN, else)</i>	returnează <i>then1</i> dacă <i>value</i> este egală cu <i>if1</i> , <i>then2</i> dacă <i>value</i> este egală cu <i>if2</i> etc.; dacă <i>value</i> nu este egală cu nici una din valorile <i>if</i> , atunci funcția întoarce valoarea <i>else</i> ;	<i>DECODE('a', 'a', 'b', 'c') = 'b'</i> <i>DECODE('b', 'a', 'b', 'c') = 'c'</i> <i>DECODE('c', 'a', 'b', 'c') = 'c'</i>

<i>NVL(expr_1, expr_2)</i>	dacă <i>expr_1</i> este <i>NULL</i> , întoarce <i>expr_2</i> ; altfel, întoarce <i>expr_1</i> . Tipurile celor două expresii trebuie să fie compatibile sau <i>expr_2</i> să poată fi convertit implicit la <i>expr_1</i>	<i>NVL(NULL, 1) = 1</i> <i>NVL(2, 1) = 2</i> <i>NVL('a', 1) = 'a' -- conversie implicită</i> <i>NVL(1, 'a') -- eroare --nu are loc conversia implicită</i>
<i>NVL2(expr_1, expr_2, expr_3)</i>	dacă <i>expr_1</i> este <i>NOT NULL</i> , întoarce <i>expr_2</i> , altfel întoarce <i>expr_3</i>	<i>NVL2(1, 2, 3) = 2</i> <i>NVL2(NULL, 1, 2) = 2</i>
<i>NULLIF(expr_1, expr_2)</i>	Dacă <i>expr_1 = expr_2</i> atunci funcția returnează <i>NULL</i> , altfel returnează expresia <i>expr_1</i> . Echivalent cu <i>CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END</i>	<i>NULLIF(1, 2) = 1</i> <i>NULLIF(1, 1) = NULL</i>
<i>COALESCE(expr_1, expr_2, ..., expr_n)</i>	Returnează prima expresie <i>NOT NULL</i> din lista de argumente.	<i>COALESCE(NULL, NULL, 1, 2, NULL) = 1</i>
<i>UID, USER</i>	întorc <i>ID</i> -ul, respectiv <i>username</i> -ul utilizatorului <i>ORACLE</i> curent	<i>SELECT USER</i> <i>FROM dual;</i>
<i>VSIZE(expr)</i>	întoarce numărul de octeți ai unei expresii de tip <i>DATE</i> , <i>NUMBER</i> sau <i>VARCHAR2</i>	<i>SELECT VSIZE(salary)</i> <i>FROM employees</i> <i>WHERE employee_id=200;</i>

Utilizarea funcției *DECODE* este echivalentă cu utilizarea clauzei *CASE* (într-o comandă SQL). O formă a acestei clauze este:

<pre> CASE expr WHEN expr_1 THEN valoare_1 [WHEN expr_2 THEN valoare_2 ... WHEN expr_n THEN valoare_n] [ELSE valoare] END </pre>	<p>În funcție de valoarea expresiei <i>expr</i> returnează <i>valoare_i</i> corespunzătoare primei clauze <i>WHEN .. THEN</i> pentru care <i>expr = expresie_i</i>; dacă nu corespunde cu nici o clauză <i>WHEN</i> atunci returnează valoarea din <i>ELSE</i>. Nu se poate specifica <i>NULL</i> pentru toate valorile de returnat. Toate valorile trebuie să aibă același tip de date.</p>
---	--

2. Funcțiile multiple-row (agregat) pot fi utilizate pentru a returna informația corespunzătoare fiecăruia dintre grupurile obținute în urma divizării liniilor tabelului cu ajutorul clauzei *GROUP BY*. Ele pot apărea în clauzele *SELECT*, *ORDER BY* și *HAVING*. *Server-ul Oracle* aplică aceste funcții fiecărui grup de linii și returnează un singur rezultat pentru fiecare mulțime.

Dintre funcțiile grup definite în sistemul *Oracle*, se pot enumera: *AVG*, *SUM*, *MAX*, *MIN*, *COUNT*, *STDDEV*, *VARIANCE* etc. Tipurile de date ale argumentelor funcțiilor grup pot fi *CHAR*, *VARCHAR2*, *NUMBER* sau *DATE*. Funcțiile *AVG*, *SUM*, *STDDEV* și *VARIANCE* operează numai asupra valorilor numerice. Funcțiile *MAX* și *MIN* pot opera asupra valorilor numerice, caracter sau dată calendaristică.

Toate funcțiile grup, cu excepția lui *COUNT(*)*, ignoră valorile *null*. *COUNT(expresie)* returnează numărul de linii pentru care expresia dată nu are valoarea *null*. Funcția *COUNT* returnează un număr mai mare sau egal cu zero și nu întoarce niciodată valoarea *null*.

Când este utilizată clauza *GROUP BY*, *server-ul* sortează implicit mulțimea rezultată în ordinea crescătoare a valorilor coloanelor după care se realizează gruparea.

II. [Join]

Join-ul este operația de regăsire a datelor din două sau mai multe tabele, pe baza valorilor comune ale unor coloane. De obicei, aceste coloane reprezintă cheia primară, respectiv cheia externă a tabelelor.

Condiția de *join* se poate scrie în clauza *WHERE* a instrucțiunii *SELECT*. Într-o instrucțiune *SELECT* care unește tabele prin operația de *join*, se recomandă ca numele coloanelor să fie precedate de numele sau alias-urile tabelelor pentru claritate și pentru îmbunătățirea timpului de acces la baza de date. Dacă același nume de coloană apare în mai mult de două tabele, atunci numele coloanei se prefixează **obligatoriu** cu numele sau *alias*-ul tabelului corespunzător. Pentru a realiza un *join* între ***n* tabele**, va fi nevoie de cel puțin ***n* – 1 condiții de join**.

Inner join (equijoin, join simplu) – corespunde situației în care valorile de pe coloanele ce apar în condiția de *join* trebuie să fie egale.

Operația va fi reluată și completată în cadrul laboratorului 3.

III. [Exerciții]

[Funcții pe șiruri de caractere]

1. Scrieți o cerere care are următorul rezultat pentru fiecare angajat:

```
<prenume angajat> <nume angajat> castiga <salariu> lunar dar doreste <salariu de 3 ori mai mare>.
```

 Etichetați coloana "Salariu ideal". Pentru concatenare, utilizați atât funcția *CONCAT* cât și operatorul "||".

```
SELECT CONCAT(CONCAT(...)) || ' castiga ' || salary
      || ... "Salariu ideal"
FROM employees;
```

2. Scrieți o cerere prin care să se afișeze prenumele salariatului cu prima litera majusculă și toate celelalte litere minuscule, numele acestuia cu majuscule și lungimea numelui, pentru angajații al căror nume începe cu J sau M sau care au a treia literă din nume A. Rezultatul va fi ordonat descrescător după lungimea numelui. Se vor eticheta coloanele corespunzător. Se cer 2 soluții (cu operatorul *LIKE* și funcția *SUBSTR*).
3. Să se afișeze, pentru angajații cu prenumele „Steven”, codul și numele acestora, precum și codul departamentului în care lucrează. Căutarea trebuie să nu fie *case-sensitive*, iar eventualele *blank*-uri care preced sau urmează numelui trebuie ignorate.

```
SELECT ...
FROM ...
WHERE LTRIM(RTRIM(UPPER(first_name)))='STEVEN';
```

sau

```
SELECT ...
FROM ...
WHERE TRIM(BOTH FROM UPPER(first_name))='STEVEN';
```

4. Să se afișeze pentru toți angajații al căror nume se termină cu litera 'e', codul, numele, lungimea numelui și poziția din nume în care apare prima dată litera 'a'. Utilizați *alias*-uri corespunzătoare pentru coloane.

[Funcții aritmetice]

5. Să se afișeze detalii despre salariații care au lucrat un număr întreg de săptămâni până la data curentă.

Obs: Soluția necesită rotunjirea diferenței celor două date calendaristice. De ce este necesar acest lucru?

6. Să se afișeze codul salariatului, numele, salariul, salariul mărit cu 15%, exprimat cu două zecimale și numărul de sute al salariului nou rotunjit la 2 zecimale. Etichetați ultimele două coloane "Salariu nou", respectiv "Numar sute". Se vor lua în considerare salariații al căror salariu nu este divizibil cu 1000.

```
SELECT employee_id, first_name, salary, ROUND(salary* 1.15, 2) "Salariu nou",
       ROUND(salary*1.15/100, 2) "Numar sute"
FROM employees
WHERE MOD(salary, 1000)!=0;
```

7. Să se listeze numele și data angajării salariaților care câștigă comision. Să se eticheteze coloanele „Nume angajat”, „Data angajării”. Utilizați funcția *RPAD* pentru a determina ca data angajării să aibă lungimea de 20 de caractere.

```
SELECT last_name AS "Nume angajat",
       RPAD(TO_CHAR(hire_date),20,' ') "Data angajarii"
FROM employees
WHERE commission_pct IS NOT NULL;
```

[Funcții și operații cu date calendaristice]

8. Să se afișeze data (numele lunii, ziua, anul, ora, minutul și secunda) de peste 30 zile.

```
SELECT TO_CHAR(SYSDATE+30, 'MONTH DD HH24:MM:SS') "Data"
FROM DUAL;
```

9. Să se afișeze numărul de zile rămase până la sfârșitul anului.

10. a) Să se afișeze data de peste 12 ore.

```
SELECT TO_CHAR(SYSDATE+12/24, 'DD/MM HH24:MM:SS') "Data"
FROM DUAL;
```

- b) Să se afișeze data de peste 5 minute

Obs: Cât reprezintă 5 minute dintr-o zi?

11. Să se afișeze numele și prenumele angajatului (într-o singură coloană), data angajării și data negocierii salariului, care este prima zi de Luni după 6 luni de serviciu. Etichetați această coloană "Negociere".

```
SELECT first_name || ' ' || last_name "Nume si prenume", hire_date,
       NEXT_DAY(ADD_MONTHS(hire_date, 6)) "Negociere"
FROM employees;
```

12. Pentru fiecare angajat să se afișeze numele și numărul de luni de la data angajării. Etichetați coloana "Luni lucrate". Să se ordoneze rezultatul după numărul de luni lucrate. Se va rotunji numărul de luni la cel mai apropiat număr întreg.

```
...
ORDER BY MONTHS_BETWEEN(SYSDATE, hire_date);
```

Sau

...

ORDER BY "Luni lucrate";

Sau

...

ORDER BY 2;

Obs: În clauza *ORDER BY*, precizarea criteriului de ordonare se poate realiza și prin indicarea *alias*-urilor coloanelor sau a pozițiilor acestora în clauza *SELECT*.

13. Să se afișeze numele, data angajării și ziua săptămânii în care a început lucrul fiecare salariat. Etichetați coloana "Zi". Ordonăți rezultatul după ziua săptămânii, începând cu Luni.

```
SELECT first_name || ' ' || last_name, hire_date, TO_CHAR(hire_date, 'Day') "Zi"
FROM employees
ORDER BY TO_CHAR(hire_date - 1, 'D');
```

[Funcții diverse]

14. Să se afișeze numele angajaților și comisionul. Dacă un angajat nu câștigă comision, să se scrie "Fara comision". Etichetați coloana "Comision".

```
SELECT first_name || ' ' || last_name,
NVL(TO_CHAR(commission_pct), 'Fara comision')
FROM employees;
```

15. Să se listeze numele, salariul și comisionul tuturor angajaților al căror venit lunar (salariu + valoare comision) depășește 10000.

```
SELECT last_name, salary, commission_pct, salary + salary * NVL(commission_pct, 0)
    venit_lunar
FROM employees
WHERE salary + salary * NVL(commission_pct, 0) > 10000;
```

[Instrucțiunea CASE, comanda DECODE]

16. Să se afișeze numele, codul job-ului, salariul și o coloană care să arate salariul după mărire. Se presupune că pentru IT_PROG are loc o mărire de 20%, pentru SA_REP creșterea este de 25%, iar pentru SA_MAN are loc o mărire de 35%. Pentru ceilalți angajați nu se acordă mărire. Să se denumească coloana "Salariu renegociat".

```
SELECT last_name, job_id, salary,
    DECODE(job_id,
        'IT_PROG', salary*1.2,
        ...
        salary) "Salariu renegociat"
FROM employees;
```

sau

```
SELECT last_name, job_id, salary,
    CASE job_id WHEN 'IT_PROG' THEN salary* 1.2
        ...
        ELSE salary
    END "Salariu renegociat"
FROM employees;
```


[Join]

17. Să se afișeze numele salariatului, codul și numele departamentului pentru toți angajații.

```
SELECT ____, employees.department_id, ____
FROM employees, departments
WHERE employees.department_id=departments.department_id;
sau
SELECT ____, e.department_id, ____
FROM employees e, departments d
WHERE e.department_id=d.department_id;
```

Obs: Am realizat operația de join între tabelele *employees* și *departments*, pe baza coloanei comune *department_id*. Observați utilizarea *alias*-urilor. Ce se întâmplă dacă eliminăm condiția de *join*?

Obs: Numele sau *alias*-urile tabelelor sunt obligatorii în dreptul coloanelor care au același nume în mai multe tabele. Altfel, nu sunt necesare dar este recomandată utilizarea lor pentru o mai bună claritate a cererii.

18. Să se listeze codurile și denumirile job-urilor care există în departamentul 30.

```
SELECT DISTINCT e.job_id, job_title
FROM jobs j, employees e
WHERE j.job_id = e.job_id AND department_id=30;
```

19. Să se afișeze numele angajatului, numele departamentului și orașul pentru toți angajații care câștigă comision.

```
SELECT ____, ____, ____
FROM ____, ____, ____
WHERE ____ AND commission_pct ____ ;
```

20. Să se afișeze numele salariatului și numele departamentului pentru toți salariații care au litera A inclusă în nume.

21. Să se afișeze numele, titlul job-ului și denumirea departamentului pentru toți angajații care lucrează în Oxford.

22. Să se afișeze codul angajatului și numele acestuia, împreună cu numele și codul șefului său direct. Se vor eticheta coloanele Ang#, Angajat, Mgr#, Manager.

```
SELECT e.employee_id Ang#, e.last_name Angajat,
       e.manager_id Mgr#, m.last_name Manager
FROM employees e, employees m
WHERE e.manager_id = m.employee_id;
```

Obs: Am realizat operația de *self-join* (inner join al tabelului cu el însuși).

23. Să se modifice cererea anterioară pentru a afișa toți salariații, inclusiv cei care nu au șef.

```
SELECT e.employee_id Ang#, e.last_name Angajat,
       e.manager_id Mgr#, m.last_name Manager
FROM employees e, employees m
WHERE e.manager_id = m.employee_id(+);
```

Obs: Am realizat operația de outer-join, indicată în SQL prin "(+)" plasat la dreapta coloanei deficitare în informație.

24. Scrieți o cerere care afișează numele angajatului, codul departamentului în care acesta lucrează și numele colegilor săi de departament. Se vor eticheta coloanele corespunzător.

```
SELECT e.last_name, e.department_id, c.last_name
```

```
FROM employees e, employees c  
WHERE e.department_id = c.department_id AND e.employee_id > c.employee_id;
```

25. Creați o cerere prin care să se afișeze numele, codul job-ului, titlul job-ului, numele departamentului și salariul angajaților. Se vor include și angajații al căror departament nu este cunoscut.

26. Să se afișeze numele și data angajării pentru salariații care au fost angajați după Gates.

```
...  
WHERE LOWER(g.last_name)='gates' AND e.hire_date>g.hire_date;
```

27. Să se afișeze numele salariatului și data angajării împreună cu numele și data angajării șefului direct pentru salariații care au fost angajați înaintea șefilor lor. Se vor eticheta coloanele Angajat, Data_ang, Manager si Data_mgr.

```
WHERE e.manager_id = m.employee_id AND e.hire_date<m.hire_date;
```