Baze de date-Anul 2
Laborator 2 SQL

Funcții *SQL.* Cereri multi-relație (introducere)

I. [Funcții SQL]

Funcțiile *SQL* sunt predefinite în sistemul *Oracle* şi 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 aşteptat, sistemul converteşte implicit argumentul înainte să evalueze funcția.

Dacă o funcție *SQL* este apelată cu un argument *null*, ea returnează automat valoarea *null*. Singurele funcții care nu urmează această regulă sunt *CONCAT*, *NVL* şi *REPLACE*.

Principalele funcții *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 *SELECT*, clauzele *WHERE*, *START WITH*, *CONNECT BY* şi *HAVING*. În ceea ce priveşte tipul argumentelor asupra cărora operează şi al rezultatelor furnizate, funcțiile *single row* pot fi clasificate în clase corespunzătoare.

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

Funcție	Descriere	Exemplu conversie
TO_CHAR	converteşte (sau formatează) un număr sau o dată calendaristică în şir de caractere	*TO_CHAR(7) = ' 7' TO_CHAR(-7) = '-7' TO_CHAR (SYSDATE, 'DD/MM/YYYY') = ' 18/04/2007'*
TO_DATE	converteşte (sau formatează) un număr sau un şir de caractere în dată calendaristică	*TO_DATE('18-APR-2007','dd-mon-yyyy')*
TO_NUMBER	converteşte (sau formatează) un şir de caractere în număr	*TO_NUMBER ('-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*.

- **Funcțiile pentru prelucrarea caracterelor** sunt prezentate în următorul tabel:

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

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

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

- **Funcțiile aritmetice** *single-row* pot opera asupra:
  - unei singure valori, şi 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* şi *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
SYSDATE	întoarce data şi timpul curent	*SELECT SYSDATE FROM dual;* (de revăzut utilizarea acestei funcții împreună cu *TO_CHAR* în cadrul laboratorului 1)
ADD_MONTHS (expr_date, nr_luni)	întoarce data care este după *nr_luni* luni de la data *expr_date*;	*ADD_MONTHS('02-APR-2007', 3)* = '02-JUL-2007'.
NEXT_DAY(expr_date, day)	întoarce următoarea dată după data *expr_date*, a cărei zi a săptămânii este cea specificată prin şirul de caractere *day*	*NEXT_DAY('18-APR-2007', 'Monday') = '23-APR-2007'*

LAST_DAY(expr_date)	întoarce data corespunzătoare ultimei zile a lunii din care data *expr_date* face parte	*LAST_DAY('02-DEC-2007') = '31-DEC-2007'*
MONTHS_BETWEEN (expr_date2, expr_date1)	întoarce numărul de luni dintre cele două date calendaristice specificate. Data cea mai recentă trebuie specificată în primul argument, altfel rezultatul este negativ.	*MONTHS_BETWEEN('02-DEC-2005', '10-OCT-2002') = 37.7419355* *MONTHS_BETWEEN('10-OCT-2002', '02-DEC-2005') = -37.7419355*
TRUNC(expr_date)	întoarce data *expr_date*, dar cu timpul setat la ora 12:00 AM (miezul nopții)	*TO_CHAR(TRUNC (SYSDATE), 'dd/mm/yy HH24:MI') = '02/12/05 00:00'*
ROUND(expr_date)	dacă data *expr_date* este înainte de miezul zilei, întoarce data *d* cu timpul setat la ora 12:00 AM; altfel, este returnată data corespunzătoare zilei următoare, cu timpul setat la ora 12:00 AM	TO_CHAR(ROUND (SYSDATE*), 'dd/mm/yy hh24:mi am') = '03/12/05 00:00 AM'*
LEAST(d1, d2, …, dn), GREATEST(d1, d2, …, dn)	dintr-o listă de date calendaristice, funcțiile întorc prima, respectiv ultima dată în ordine cronologică	*LEAST(SYSDATE, SYSDATE + 3, SYSDATE – 5) = SYSDATE-5* *GREATEST(SYSDATE, SYSDATE + 3, SYSDATE – 5) = SYSDATE + 3*

Operațiile care se pot efectua asupra datelor calendaristice sunt următoarele:

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

- **Funcții diverse:**

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

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

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

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

2. Funcțiile multiple-row (agregat) pot fi utilizate pentru a returna informația corespunzătoare fiecăruia dinte grupurile obținute în urma divizării liniilor tabelului cu ajutorul clauzei GROUP BY. Ele pot apărea în clauzele *SELECT*, *ORDER BY* şi *HAVING*. *Server*-ul *Oracle* aplică aceste funcții fiecărui grup de linii şi 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* şi *VARIANCE* operează numai asupra valorilor numerice. Funcțiile *MAX* şi *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 şi 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 scrie în clauza *WHERE* a instrucțiunii *SELECT*. Într-o instrucțiune *SELECT* care uneşte tabele prin operația de *join*, se recomandă ca numele coloanelor să fie precedate de numele sau alias-urile tabelelor pentru claritate şi pentru îmbunătățirea timpului de acces la baza de date. Dacă acelaşi 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 şi completată în cadrul laboratorului 3.

III. [Exerciții]

[Funcții pe şiruri 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>*. Etichetati coloana "Salariu ideal". Pentru concatenare, utilizați atât funcția *CONCAT* cât şi operatorul "||".

```
SELECT CONCAT(CONCAT(first_name, ' '), last_name) ||' castiga '|| salary
          || ' lunar dar doreste '|| salary*3 "Salariu ideal"
FROM employees;
```

2. Scrieți o cerere prin care să se afişeze prenumele salariatului cu prima litera majusculă şi toate celelalte litere minuscule, numele acestuia cu majuscule şi 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* şi funcția *SUBSTR*).

```
SELECT INITCAP(first_name), UPPER(last_name), LENGTH(last_name) Lungime
FROM    employees
WHERE LOWER(last_name) LIKE 'j%' OR LOWER(last_name) LIKE '__a%'
          OR LOWER(last_name) LIKE 'm%'
ORDER BY 3 DESC; -- echivalent cu ORDER BY Lungime DESC
                          -- sau ORDER BY LENGTH(last_name) DESC
```

sau

```
SELECT INITCAP(first_name), UPPER(last_name), LENGTH(last_name) Lungime
FROM    employees
WHERE SUBSTR(LOWER(last_name), 1, 1) = 'j' OR
          SUBSTR(LOWER(last_name), 3, 1) = 'a' OR
          SUBSTR(LOWER(last_name), 1,1) = 'm'
ORDER BY 3 DESC;
```

3. Să se afişeze pentru angajații cu prenumele „Steven", codul, numele şi codul departamentului în care lucrează. Căutarea trebuie să nu fie *case-sensitive*, iar eventualele *blank*-uri care preced sau urmează numelui trebuie ignorate.

```
SELECT employee_id,  last_name,  department_id
FROM employees
```

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

sau

```
SELECT employee_id, last_name,  department_id
FROM employees
WHERE TRIM(BOTH FROM UPPER(first_name))='STEVEN';
```

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

```
SELECT employee_id, last_name,  LENGTH(last_name) lungime,
        INSTR(LOWER(last_name), 'a') pozitie
FROM employees
WHERE LOWER(SUBSTR(last_name,-1))='e';
```

[Funcții aritmetice]

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

```
SELECT employee_id, last_name, salary
FROM employees
WHERE MOD(ROUND(SYSDATE – hire_date), 7)=0;
```

De ce este necesară rotunjirea diferentei celor două date calendaristice?

6. Să se afişeze codul salariatului, numele, salariul, salariul mărit cu 15%, exprimat cu două zecimale şi 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. Salvați instrucțiunea *SQL* într-un fişier *p2l2.sql*.

```
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;
```

```
SAVE p6l2.sql
```

7. Rulați fişierul *p6l2.sql*.

8. Să se modifice *p6l2.sql* pentru a adauga o nouă coloană, care va scade salariul vechi din salariul nou, rezultatul fiind afişat în sute.

```
SELECT employee_id, first_name, salary, ROUND(salary* 1.15, 2) “Salariu nou”,
        ROUND (salary*0.15/100, 2) “Diferenta in sute”
FROM employees;
```

9. Să se listeze numele şi data angajării salariaților care câştigă comision. Să se eticheteze coloanele „Nume angajat”, „Data angajarii”. Pentru a nu obține *alias*-ul datei angajării trunchiat, utilizați funcția *RPAD*.

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

[Funcții şi operații cu date calendaristice]

10. Să se afişeze data (numele lunii, ziua, anul, ora, minutul si secunda) de peste 30 zile.
SELECT TO_CHAR(SYSDATE+30, 'MONTH DD HH24:MM:SS') "Data"
FROM DUAL;

11. Să se afişeze numărul de zile rămase până la sfârşitul anului.
SELECT TO_DATE('31-DEC-2007')-SYSDATE "Days to go"
FROM DUAL;

12. a) Să se afişeze data de peste 12 ore.
SELECT TO_CHAR(SYSDATE+12/24, ' DD/MM HH24:MM:SS') "Data"
FROM DUAL;

b) Să se afişeze data de peste 5 minute
SELECT TO_CHAR(SYSDATE+1/288, ' DD/MM HH24:MM:SS') "Data"
FROM DUAL;
***Obs**:* O zi reprezintă un întreg, iar 5 minute sunt a 288-a parte dintr-o zi.

13. Să se afişeze numele şi prenumele angajatului (într-o singură coloană), data angajării şi 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), 'Monday') "Negociere"
FROM employees;

14. Ce efect are comanda *EDIT* (din SQL*Plus) dacă nu se specifică nici un nume de fişier? Utilizați această comandă pentru a modifica cererea anterioară (fără salvare prealabilă în fişier SQL), considerând ca dată a negocierii prima zi de Vineri după 3 luni de serviciu.

Obs: Fără specificarea unui nume de fişier, comanda *EDIT* oferă posibilitatea modificării comenzii aflate în buffer-ul *SQL*. După modificarea comenzii din buffer, se salvează şi se închide fişierul care conține această comandă, iar executarea *buffer*-ului se realizează prin comanda "/" sau *RUN*.

15. Pentru fiecare angajat să se afişeze numele şi 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.

SELECT last_name,
ROUND(MONTHS_BETWEEN(SYSDATE, hire_date)) "Luni lucrate"
FROM employees
ORDER BY MONTHS_BETWEEN(SYSDATE, hire_date);
Sau
SELECT last_name,
ROUND(MONTHS_BETWEEN(SYSDATE, hire_date)) "Luni lucrate"
FROM employees
ORDER BY "Luni lucrate";
Sau
SELECT last_name,
ROUND(MONTHS_BETWEEN(SYSDATE, hire_date)) "Luni lucrate"
FROM employees
ORDER BY 2;

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

16. Să se afişeze numele, data angajării şi ziua săptămânii în care a început lucrul fiecare salariat. Etichetați coloana "Zi". Ordonați rezultatul după ziua săptămânii, începând cu Luni.

```
SELECT      last_name, hire_date, TO_CHAR(hire_date, 'day') Zi
FROM        employees
ORDER BY  TO_CHAR(hire_date-1, 'd');
```

[Funcții diverse]

17. Să se afişeze numele angajaților şi comisionul. Dacă un angajat nu câştigă comision, să se scrie "Fara comision". Etichetați coloana "Comision".

```
SELECT __, NVL(__, __) ___
FROM ___;
```

18. Să se listeze numele, salariul şi comisionul tuturor angajaților al căror venit lunar depăşeşte 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]

19. Să se afişeze numele, codul job-ului, salariul şi o coloană care să arate salariul după mărire. Se presupune că pentru IT_PROG are loc o mărire de 20%, pentru SA_REP creşterea 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,
              'SA_REP', salary*1.25,
              'SA_MAN', salary*1.35,
              salary) "Salariu renegociat"
FROM employees;
```

sau

```
SELECT last_name, job_id, salary,
  CASE job_id WHEN 'IT_PROG' THEN salary* 1.2
                WHEN 'SA_REP' THEN salary*1.25
               WHEN 'SA_MAN' THEN salary*1.35
                ELSE salary
  END "Salariu renegociat"
FROM employees;
```

[Join]

20. Să se afişeze numele salariatului, codul şi 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* şi *departments*, pe baza câmpului comun *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 acelaşi nume în mai multe tabele. Altfel, nu sunt necesare dar este recomandată utilizarea lor pentru o mai bună claritate a cererii.

21. Să se listeze job-urile 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;
```

22. Să se afişeze numele angajatului, numele departamentului şi locatia pentru toți angajații care câştigă comision.

```
SELECT ____, _____, _____
FROM   ____, _____
WHERE_____ AND  commission_pct _____ ;
```

23. Să se afişeze numele salariatului şi numele departamentului pentru toți salariații care au litera A inclusă în nume.

```
SELECT  last_name, department_name
FROM     employees e, departments d
WHERE  e.department_id=d.department_id AND lower(last_name) LIKE '%a%';
```

24. Să se afişeze numele, job-ul, codul şi numele departamentului pentru toți angajații care lucrează în Oxford.

```
SELECT last_name, job_id, e.department_id, department_name
FROM     employees e, departments d, locations l
WHERE  e.department_id = d.department_id AND d.location_id = l.location_id
          AND lower(city)='oxford';
```

25. Să se afişeze codul angajatului şi numele acestuia, împreună cu numele şi codul şefului său direct. Se vor eticheta coloanele Ang#, Angajat, Mgr#, Manager. Să se salveze instrucțiunea într-un fişier numit *p17l2.sql*.

```
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;

SAVE p25l2
```

Obs: Am realizat operația de self-join (join al unui tabel cu el însuşi).

26. Să se modifice *p25l2.sql* pentru a afişa toți salariații, inclusiv cei care nu au şef. Salvati ca *p26l2.sql*. Rulați p26l2.sql.

Cererea modificată va arăta astfel:

```
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.

27. Sa se listeze numele, salariul si comisionul tuturor angajatilor al căror salariu total (cu tot cu comision) depăşeste 10000$.

```
SELECT last_name, first_name, salary, commision_pct,
```

```
        salary + salary * NVL(commision_pct, 0)
FROM employees
WHERE salary + salary * NVL(commision_pct, 0) > 10000;
```

28. Creați o cerere care să afişeze numele angajatului, codul departamentului şi toți salariații care lucrează în acelaşi departament cu el. 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;
```

29. Să se listeze structura tabelului JOBS. Creați o cerere prin care să se afişeze numele, codul job-ului, titlul job-ului, numele departamentului şi salariul angajaților.

```
DESC jobs
SELECT last_name, e.job_id, job_title, department_name, salary
FROM    employees e, jobs j, departments d
WHERE  e.department_id = d.department_id AND e.job_id = j.job_id;
```

30. Să se afişeze numele şi data angajării pentru salariații care au fost angajați după *Gates*.

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

31. Să se afişeze numele salariatului şi data angajării împreună cu numele şi data angajării şefului direct pentru salariații care au fost angajați înaintea şefilor lor. Se vor eticheta coloanele Angajat, Data_ang, Manager si Data_mgr.

```
SELECT e.last_name Angajat, e.hire_date Data_ang, m.last_name Manager,
        m.hire_date Data_mgr
FROM    employees e, employees m
WHERE  e.manager_id = m.employee_id AND e.hire_date<m.hire_date;
```