

// lab 2 26 - 02 - 2020

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 "||".

/* ex1 */

```
SELECT first_name , last_name || ' castiga ' || salary || ' lunar, dar doreste ' || salary * 3 as "
SALARIU IDEAL "
FROM EMPLOYEES;
```

/* ex 2 */

Scrieți o cerere prin care să se afișeze prenumele salariatului cu prima litera majusculă și toate celelalte litere mici, 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).

```
SELECT INITCAP(first_name) as "prenume", UPPER(last_name) as "nume",
LENGTH(last_name) as "lungime"
FROM EMPLOYEES
WHERE UPPER(last_name) LIKE ('J%') OR UPPER(last_name) LIKE ('M%') OR
UPPER(last_name) LIKE ('__A%')
ORDER BY 3 DESC;
```

```
SELECT INITCAP(first_name) as "prenume", UPPER(last_name) as "nume",
LENGTH(last_name) as "lungime"
FROM EMPLOYEES
WHERE SUBSTR( UPPER(last_name), 1,1) = 'J' OR SUBSTR(UPPER(last_name), 1 ,
1)= 'M' OR SUBSTR( UPPER(last_name), 3 ,1 ) = 'A'
ORDER BY 3 DESC;
```

/* Ex3 */

Să se afișeze pentru angajații cu prenumele „Steven”, codul, numele 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 EMPLOYEES;
SELECT employee_id , last_name , department_id
FROM employees
```

```
WHERE UPPER(TRIM(first_name)) LIKE 'STEVEN';
```

```
/* EX4 */
```

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.

```
SELECT employee_id,last_name,LENGTH(last_name) "lungime  
nume",INSTR(LOWER(last_name),'a',1) "prima aparitie a"  
FROM employees  
WHERE UPPER(last_name) LIKE '%E';
```

```
/*EX5*/
```

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

```
SELECT *  
FROM employees  
WHERE MOD(ROUND(SYSDATE - hire_date), 7) = 0;
```

```
/* ex 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, last_name, salary, ROUND(salary*1.15, 2) as "salariu nou",  
ROUND(salary*1.15, 2)/100 as "numar sute"  
FROM EMPLOYEES  
WHERE MOD(salary, 1000) <> 0;
```

```
/* EX7 */
```

7. Să se listeze numele și data angajării salariaților care câștigă 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(hire_date, 20, '&') AS "data angajarii"  
FROM EMPLOYEES  
WHERE commission_pct IS NOT NULL;
```

*/ EX8 */

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

```
SELECT TO_CHAR(SYSDATE, 'mm-dd-yyyy hh:mi:ss') "data curenta",  
TO_CHAR(SYSDATE+30, 'mm-dd-yyyy hh:mi:ss') "data + 30"  
FROM DUAL;
```

/* EX9 */

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

```
SELECT ROUND(TO_DATE('31-DEC-2020', 'dd-mon-yyyy') - SYSDATE)  
FROM DUAL;
```

/* EX10 */

10. a) Să se afișeze data de peste 12 ore.
b) Să se afișeze data de peste 5 minute.

```
SELECT SYSDATE + 0.5  
FROM DUAL;
```

```
SELECT SYSDATE +INTERVAL '12' HOUR FROM DUAL;
```

```
SELECT TO_CHAR(SYSDATE, 'dd-mm-yyyy "azi este cald" ' )  
FROM DUAL;
```

```
SELECT SYSDATE + 5 / (60*24)  
FROM DUAL;
```

/* EX11 */

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 last_name || ' ' || first_name, hire_date, NEXT_DAY(ADD_MONTHS(hire_date, 6),  
'monday') "negociere"  
FROM EMPLOYEES;
```

/* EX12 */

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.

```
SELECT last_name, ROUND(MONTHS_BETWEEN(SYSDATE, hire_date), 2) "luni lucrate"
FROM EMPLOYEES;
```

/* EX13 */

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 last_name, hire_date, TO_CHAR(hire_date, 'd') "zi"
FROM EMPLOYEES
ORDER BY MOD(TO_CHAR(hire_date+6, 'd'), 7);
```

4 MARTIE 2020

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

```
SELECT last_name, NVL(TO_CHAR(commission_pct), 'Fara comision') "Comision"
FROM employees;
```

--var 2

```
SELECT last_name, NVL2(TO_CHAR(commission_pct), TO_CHAR(commission_pct), 'Fara
comision') "Comision"
FROM employees;
```

--var 3

```
SELECT last_name, DECODE(TO_CHAR(commission_pct), NULL, 'Fara comision',
commission_pct) "Comision"
FROM employees;
```

15. Să se listeze numele, salariul și comisionul tuturor angajaților al căror venit lunar depășește 10000\$.

```
SELECT last_name, salary, commission_pct
FROM employees
WHERE salary*(1+NVL(commission_pct,0))>10000;
```

/ex 16/

16. Sa 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(UPPER(job_id), 'IT_PROG', salary*1.2,
'SA_REP', salary*1.25, 'SA_MAN', salary*1.35, salary) "Salariu renegociat"
FROM employees;
```

```
SELECT last_name, job_id, salary,
CASE UPPER(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;
```

/* ex17 */

17. Să se afișeze numele salariatului, codul și numele departamentului pentru toți angajații. Obs: Numele sau alias-urile tabelor 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.

--var standard

```
SELECT last_name, department_id, department_name
FROM employees
JOIN departments USING(department_id);
```

--var cu clauza on

```
SELECT last_name, e.department_id, department_name
FROM employees e
JOIN departments d ON(e.department_id = d.department_id);
```

--var non-standard

```
SELECT last_name, e.department_id, department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id;
```

/*ex 18*/

18. Să se listeze titlurile job-urile care există în departamentul 30.

```
SELECT job_title
FROM jobs
JOIN employees USING (job_id)
WHERE department_id=30;
```

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

```
SELECT last_name, department_name, city
FROM employees
JOIN departments USING(department_id)
JOIN locations USING(location_id)
WHERE commission_pct
IS NOT NULL;
```

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

*/

```
SELECT last_name , department_name
FROM employees e
JOIN departments d ON(e.department_id = d.department_id AND
UPPER(TRIM(last_name)) LIKE '%A%');ang
```

/* ex 21 */

Să se afișeze numele, job-ul, codul și numele departamentului pentru toți angajații care lucrează în Oxford.

```
SELECT last_name , job_id , department_id, department_name
FROM employees
JOIN departments USING (department_id)
JOIN locations USING(location_id)
WHERE INITCAP(city) LIKE '%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 ang.employee_id AS "Ang#", ang.last_name AS "Angajat", sef.employee_id AS
"Mgr#" , sef.last_name AS "Manager"
FROM employees ang
JOIN employees sef ON (ang.manager_id=sef.employee_id);
```

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

```
SELECT ang.employee_id AS "Ang#", ang.last_name AS "Angajat", sef.employee_id AS  
"Mgr#" , sef.last_name AS "Manager"  
FROM employees ang  
LEFT JOIN employees sef ON (ang.manager_id=sef.employee_id);
```

24. Creați o cerere care să afișeze numele angajatului, codul departamentului și toți salariații care lucrează în același departament cu el. Se vor eticheta coloanele corespunzător.

```
SELECT ang.last_name, ang.department_id, coleg.last_name  
FROM employees ang  
JOIN employees coleg ON(ang.department_id=coleg.department_id AND ang.employee_id  
< coleg.employee_id)  
ORDER BY 2;
```

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

```
SELECT e.last_name, e.hire_date, g.last_name, g.hire_date  
FROM employees e  
JOIN employees g ON(e.hire_date > g.hire_date AND INITCAP(g.last_name) LIKE  
'%Gates%');
```

--11.03.2020 nu s-a tinut datorita COVID-19
lab3 tema!

--18.03.2020

--ex2

```
SELECT MAX(salary) "Maxim", MIN(salary) "Minim", SUM(salary) "Suma",  
ROUND(AVG(salary),2) "Media"  
FROM employees;
```

--ex3

```
SELECT MAX(salary) "Maxim", MIN(salary) "Minim", SUM(salary) "Suma",  
ROUND(AVG(salary),2) "Media"  
FROM employees  
GROUP BY job_id;
```

--Exercitiul4

```
SELECT COUNT(employee_id), job_id  
FROM employees  
GROUP BY job_id;
```

-- Exercițiul 5

```
SELECT COUNT(DISTINCT MANAGER_ID)
```

```
FROM employees
WHERE manager_id IS NOT NULL
```

```
-- Exercițiul 6
SELECT MAX(salary) - MIN(salary) AS "Diferenta"
FROM employees;
```

```
-- Exercițiul 7
SELECT department_name, city, count(employee_id), round(avg(salary))
FROM employees
JOIN DEPARTMENTS USING (DEPARTMENT_ID)
JOIN locations USING (location_id)
GROUP BY department_name, city
```

```
-- Exercițiul 8
SELECT employee_id, last_name, salary
FROM employees
WHERE salary > (
    SELECT AVG(salary) FROM employees
)
ORDER BY salary DESC
```

```
-- Exercițiul 9
SELECT manager_id, MIN(salary)
FROM employees
WHERE manager_id IS NOT NULL
GROUP BY manager_id
HAVING MIN(salary) >= 3000
```

```
Dacă vrem detalii despre subordonatul cu salariul minim:
SELECT ang.employee_id, ang.last_name, ang2.manager_id, minim
FROM (
    SELECT manager_id, MIN(salary) AS minim
    FROM employees
    WHERE manager_id IS NOT NULL
    GROUP BY manager_id
    HAVING MIN(salary) >= 3000
) ang2
JOIN employees ang ON ang.manager_id = ang2.manager_id
WHERE ang.salary = minim
```

```
/* ex 10 */
```

```
SELECT department_id, department_name, MAX(salary)
FROM employees
```



```
JOIN departments USING (department_id)
GROUP BY department_id, department_name
HAVING MAX(salary) > 3000;
```

```
/* ex 11 */
```

```
SELECT MIN(AVG(SALARY))
FROM employees
GROUP BY job_id;
```

```
/* ex 12 */
```

```
SELECT department_id , department_name , SUM(salary)
FROM employees
JOIN departments USING(department_id)
GROUP BY department_name, department_id;
```

```
/* ex 13 */
```

```
SELECT ROUND(MAX(AVG(salary)),2)
FROM employees
GROUP BY department_id;
```

```
SELECT MAX(ROUND(AVG(salary),2))
FROM employees
GROUP BY department_id;
```

```
/* ex 14 */
```

```
SELECT job_id , job_title , ROUND(AVG(salary),2)
FROM employees
JOIN jobs USING(job_id)
GROUP BY job_id, job_title
HAVING ROUND(AVG(salary),2) = (
    SELECT ROUND(MIN(AVG(salary)) ,2)
    FROM employees
    GROUP BY job_id
);
```

```
/* ex 15 */
```

```
SELECT AVG(salary)
FROM employees
HAVING AVG(salary) > 2500;
```

```
/*16. Să se afișeze suma salariilor pe departamente și, în cadrul acestora, pe job-uri.*/
```

```
SELECT SUM(salary)
FROM employees
GROUP BY department_id , job_id;
```

```
SELECT SUM(salary)
FROM employees
GROUP BY job_id , department_id;
```

/*17. Să se afișeze numele departamentului si cel mai mic salariu din departamentul avand cel mai mare salariu mediu.*/

--nu e corect

```
SELECT department_name , MIN(salary)
FROM employees
JOIN departments USING(department_id)
GROUP BY department_name, salary
HAVING salary = (SELECT MAX(AVG(salary))
                  FROM employees
                  GROUP BY department_id);
```

```
SELECT MIN(salary)
FROM employees
GROUP BY department_id;
```

```
SELECT d.department_name , MIN(e.salary)
FROM employees e
JOIN departments d ON(e.department_id = d.department_id)
JOIN (SELECT department_id , AVG(salary) AS medie
      FROM employees
      GROUP BY department_id) aux ON(aux.department_id = e.department_id)
WHERE medie = (SELECT MAX(AVG(salary))
              FROM employees
              GROUP BY department_id)
GROUP BY department_name;
```

/*18.Sa se afiseze codul, numele departamentului si numarul de angajati care lucreaza in acel departament pentru:

a) departamentele in care lucreaza mai putin de 4 angajati;

b) departamentul care are numarul maxim de angajati.*/

--a

```
SELECT d.department_id , department_name , COUNT(e.employee_id)
FROM employees e
JOIN departments d ON(e.department_id = d.department_id)
GROUP BY d.department_id , department_name
HAVING COUNT(*) < 4;
```

```
--b
SELECT d.department_id , department_name , COUNT(e.employee_id)
FROM employees e
JOIN departments d ON(e.department_id = d.department_id)
GROUP BY d.department_id , department_name
HAVING COUNT(*) = (SELECT MAX(COUNT(employee_id))
                    FROM employees
                    GROUP BY department_id);
```

/*19. Sa se afiseze salariatii care au fost angajati în aceeași zi a lunii în care cei mai multi dintre salariați au fost angajati.*/

```
SELECT last_name
FROM employees
WHERE TO_CHAR(hire_date , 'DD') = (
SELECT TO_CHAR(hire_date , 'DD')
FROM employees
GROUP BY TO_CHAR(hire_date , 'DD')
HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                    FROM employees
                    GROUP BY TO_CHAR(hire_date , 'DD')));
```

--25.03.2020

lab4

--EX 20 --

```
SELECT COUNT(COUNT(department_id))
FROM employees
GROUP BY department_id
HAVING COUNT(employee_id)>15;
```

-- EX 21 --

```
SELECT department_id, SUM(salary)
FROM employees
GROUP BY department_id
HAVING COUNT(employee_id)>10 and department_id <>30;
```

-- EX 22 --

```
SELECT e.department_id, department_name, nr , medie, last_name, salary, job_id
FROM employees e
RIGHT JOIN departments d ON (e.department_id=d.department_id)
LEFT JOIN (SELECT department_id, COUNT(employee_id) nr, ROUND(AVG(salary))
           medie
```

```
FROM employees GROUP BY department_id) aux ON  
(d.department_id=aux.department_id);
```

-- EX 23 --

```
SELECT city,department_name, job_id, SUM(salary)  
FROM departments  
JOIN locations USING (location_id)  
JOIN employees USING (department_id)  
GROUP BY city, department_name, job_id,department_id  
HAVING department_id>80;
```

-- sau --

-- EX 24 --

```
SELECT last_name  
FROM employees  
WHERE employee_id IN (  
SELECT employee_id  
FROM job_history  
GROUP BY employee_id  
HAVING COUNT(job_id)>1);
```

-- sau --

```
SELECT last_name  
FROM employees e  
JOIN (SELECT employee_id, COUNT(job_id) nr  
FROM job_history  
GROUP BY employee_id) aux ON (e.employee_id=aux.employee_id)  
WHERE nr>1;
```

-- EX 25 --

```
SELECT AVG(NVL(commission_pct, 0))  
FROM employees;
```

```
SELECT SUM(commission_pct)/COUNT(*)  
FROM employees;
```

-- EX 26 --

-- EX 27 --

```
SELECT job_id AS "Job", SUM(salary) AS "Total",  
(SELECT SUM(salary) FROM employees WHERE department_id =30) AS "Dep30",  
(SELECT SUM(salary) FROM employees WHERE department_id =50) AS "Dep50",  
(SELECT SUM(salary) FROM employees WHERE department_id =80) AS "Dep80"  
FROM employees  
GROUP BY job_id;
```

-- sau --

```
SELECT job_id AS "Job", SUM(salary) AS "Total",  
SUM(DECODE(department_id, 30,salary,0)) AS "Dep30",  
SUM(DECODE(department_id, 50,salary,0)) AS "Dep50",  
SUM(DECODE(department_id, 80,salary,0)) AS "Dep80"  
FROM employees  
GROUP BY job_id;
```

-- EX 28 --

```
SELECT COUNT(employee_id),  
SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1997,1, 0)) AS "1997",  
SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1998,1, 0)) AS "1998",  
SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1999,1, 0)) AS "1999",  
SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),2000,1, 0)) AS "2000"  
FROM employees;
```

-- EX 29 --

```
SELECT e.department_id, department_name,  
(SELECT COUNT(employee_id)FROM employees WHERE  
department_id=d.department_id) AS "NR" ,  
(SELECT ROUND(AVG(salary)) FROM employees WHERE  
department_id=d.department_id) AS "MEDIE",  
last_name, salary, job_id  
FROM employees e  
RIGHT JOIN departments d ON (e.department_id=d.department_id);
```

-- EX 30 --

```
SELECT d.department_id, department_name, suma  
FROM departments d  
JOIN (SELECT department_id, SUM(salary) suma FROM employees GROUP BY  
department_id) aux
```

```
ON (D.department_id=aux.department_id);
```

```
-- EX 31--
```

```
SELECT last_name, salary, e.department_id, salariu_mediu
FROM employees e
JOIN (SELECT department_id, AVG(salary) salariu_mediu FROM employees GROUP BY
department_id) aux
ON (e.department_id=aux.department_id);
```

```
-- EX 32 --
```

```
SELECT last_name, salary, e.department_id, salariu_mediu, nr_angajati
FROM employees e
JOIN (SELECT department_id, AVG(salary) salariu_mediu, COUNT(employee_id)
nr_angajati FROM employees GROUP BY department_id) aux
ON (e.department_id=aux.department_id);
```

```
-- EX 33 --
```

```
SELECT department_name, last_name, salary
FROM employees e
JOIN departments d ON(e.department_id=d.department_id)
JOIN (SELECT department_id, MIN(salary) minim FROM employees GROUP BY
department_id) aux ON (aux.department_id=d.department_id)
WHERE salary=minim;
```

```
-- sau --
```

```
SELECT department_name, last_name, salary
FROM employees e
JOIN departments d ON(e.department_id=d.department_id)
WHERE (e.department_id, salary) IN (SELECT department_id, MIN(salary) minim FROM
employees GROUP BY department_id) ;
```

```
-- EX 34 --
```

```
SELECT e.department_id, department_name, nr , medie, last_name, salary, job_id
FROM employees e, departments d,
(SELECT department_id, COUNT(employee_id) nr, ROUND(AVG(salary)) medie
FROM employees GROUP BY department_id) aux
WHERE e.department_id(+) = d.department_id and d.department_id=aux.department_id(+);
```

```
-- exemplu ROLLUP --
```

```
SELECT department_id, TO_CHAR(hire_date, 'yyyy'), SUM(salary)
FROM employees
WHERE department_id < 50
GROUP BY ROLLUP(department_id, TO_CHAR(hire_date, 'yyyy'));
```

-- sau --

```
SELECT department_id, TO_CHAR(hire_date, 'yyyy'), SUM(salary)
FROM employees
WHERE department_id<50
GROUP BY department_id, TO_CHAR(hire_date, 'yyyy')
UNION
SELECT department_id,null, SUM(salary)
FROM employees
WHERE department_id<50
GROUP BY department_id
UNION
SELECT null ,null, SUM(salary)
FROM employees
WHERE department_id<50;
```

-- EXEMPLU CUBE --

```
SELECT department_id, TO_CHAR(hire_date, 'yyyy'), SUM(salary)
FROM employees
WHERE department_id < 50
GROUP BY CUBE (department_id, TO_CHAR(hire_date, 'yyyy'));
```

-- sau --

```
SELECT department_id, TO_CHAR(hire_date, 'yyyy'), SUM(salary)
FROM employees
WHERE department_id<50
GROUP BY department_id, TO_CHAR(hire_date, 'yyyy')
UNION
SELECT department_id,null, SUM(salary)
FROM employees
WHERE department_id<50
GROUP BY department_id
UNION
SELECT null, TO_CHAR(hire_date, 'yyyy'), SUM(salary)
FROM employees
WHERE department_id<50
GROUP BY TO_CHAR(hire_date, 'yyyy')
```

```
UNION
SELECT null ,null, SUM(salary)
FROM employees
WHERE department_id<50;
```

-- exemplu --

-- ROLLUP--

```
SELECT department_id, TO_CHAR(hire_date, 'yyyy'), SUM(salary)
FROM employees
WHERE department_id < 50
GROUP BY GROUPING SETS((department_id, TO_CHAR(hire_date,
'yyyy')), (department_id, ());
```

--CUBE--

```
SELECT department_id, TO_CHAR(hire_date, 'yyyy'), SUM(salary)
FROM employees
WHERE departmen_id < 50
GROUP BY GROUPING SETS((department_id, TO_CHAR(hire_date,
'yyyy')), (department_id, ()), (TO_CHAR(hire_date, 'yyyy')));
```

1.04.2020

-- Laborator 5 - Baze de date

-- Exercițiul 1

-- a)

```
SELECT department_name, job_title, ROUND(AVG(salary))
FROM departments
JOIN employees USING(department_id)
JOIN jobs USING(job_id)
GROUP BY ROLLUP(department_name, job_title);
```

-- b)

```
SELECT department_name, GROUPING(department_name),
job_title, GROUPING(job_title), ROUND(AVG(salary))
FROM departments
JOIN employees USING(department_id)
JOIN jobs USING(job_id)
GROUP BY ROLLUP(department_name, job_title);
```

--Exercițiul 2

-- a)

```
SELECT department_name, job_title, ROUND(AVG(salary))
FROM departments
```



```

JOIN employees USING(department_id)
JOIN jobs USING(job_id)
GROUP BY CUBE(department_name, job_title);

```

```

-- b)
SELECT department_name,
job_title, ROUND(AVG(salary)),
DECODE (GROUPING(department_name), 0, DECODE( GROUPING(job_title), 0, 'DEP
AND JOB' , 1, 'DEP'), 1, DECODE(GROUPING(job_title), 0, 'JOB', '-') "Participare"
FROM departments
JOIN employees USING(department_id)
JOIN jobs USING(job_id)
GROUP BY CUBE(department_name, job_title);

```

```

--Exercitiul 3
SELECT department_name, job_title, employees.manager_id, MAX(salary), SUM(salary)
FROM departments
JOIN employees USING(department_id)
JOIN jobs USING(job_id)
GROUP BY GROUPING SETS(( department_name, job_title) , (job_title,
employees.manager_id), ());

```

```

--Exercitiul 4
SELECT MAX(salary)
FROM employees
WHERE salary > 15000;

```

```

SELECT MAX(salary)
FROM employees
HAVING MAX(salary) > 15000;

```

-- II. Subcereri corelate

--Exercitiul 5

```

-- a)
SELECT employee_id, last_name, salary
FROM employees e
WHERE salary > (SELECT AVG(salary) FROM employees
                WHERE department_id=e.department_id AND employee_id <> e.employee_id);

```

```

-- b)
SELECT employee_id, last_name, salary, department_name,
(SELECT AVG(salary) FROM employees WHERE department_id=e.department_id) "Medie"
FROM employees e
JOIN departments d ON(d.department_id = e.department_id)
WHERE salary > (SELECT AVG(salary)FROM employees
                WHERE department_id=e.department_id AND employee_id <> e.employee_id);

```

--Exercitiul 5

```
SELECT employee_id, last_name, salary, department_name, medie
FROM employees e
JOIN (SELECT AVG(salary) medie, department_id FROM employees
GROUP BY department_id) aux ON (aux.department_id=e.department_id)
JOIN departments d ON(d.department_id = e.department_id)
WHERE salary > medie;
```

--Exercitiul 6

--var1--

```
SELECT last_name, salary
FROM employees
WHERE salary > ALL(SELECT AVG(salary) FROM employees GROUP BY department_id);
```

--VAR2--

```
SELECT last_name, salary
FROM employees
WHERE salary > (SELECT MAX(AVG(salary)) FROM employees GROUP BY
department_id);
```

--Exercitiul 7

--var 1--

```
SELECT last_name, salary
FROM employees e
WHERE salary = (SELECT MIN(salary) FROM employees WHERE
e.department_id=department_id);
```

--var 2--

```
SELECT last_name, salary
FROM employees
WHERE (salary,department_id) IN (SELECT MIN(salary), department_id FROM employees
GROUP BY department_id);
```

--var 3--

```
SELECT last_name, salary
FROM employees e
JOIN (SELECT MIN(salary) minim, department_id FROM employees GROUP BY
department_id) aux
ON ( e.department_id = aux.department_id)
WHERE salary = minim;
```

--Exercitiul 8--

```
SELECT last_name
FROM employees e
```

```
WHERE hire_date = (SELECT MIN(hire_date) FROM employees WHERE  
e.department_id=department_id);
```

--Exercitiul 9--

```
SELECT last_name  
FROM employees e  
WHERE EXISTS ( SELECT 1 FROM employees WHERE department_id=e.department_id  
AND salary = (SELECT MAX(salary) FROM employees WHERE department_id=30));
```

--Exercitiul 10--

```
SELECT last_name, salary  
FROM employees  
ORDER BY salary DESC;
```

--Nu merge--

```
SELECT last_name, salary  
FROM employees  
WHERE ROWNUM <= 5  
ORDER BY salary DESC;
```

-- varianta corecta--

```
SELECT *  
FROM ( SELECT last_name, salary  
FROM employees  
ORDER BY salary DESC)  
WHERE ROWNUM <= 5;
```

--Exercitiul 11-

```
SELECT employee_id, last_name, first_name  
FROM employees e  
WHERE (SELECT COUNT(manager_id) FROM employees WHERE  
manager_id=e.employee_id  
GROUP BY manager_id) >=2;
```

--Exercitiul 12--

--var 1--

```
SELECT city  
FROM locations l  
WHERE EXISTS( SELECT 'c' FROM departments WHERE l.location_id=location_id );
```

--var 2--

```
SELECT city  
FROM locations  
WHERE location_id IN( SELECT location_id FROM departments);
```

--tema JOIN, minus--

--Exercitiul 13--

```
SELECT department_id, department_name
FROM departments d
WHERE NOT EXISTS( SELECT -1 FROM employees WHERE
d.department_id=department_id);
```

--tema JOIN--

--III. Subcereri ierarhice

--Exercitiul 14--

-- a)

```
SELECT employee_id, last_name, hire_date, salary, manager_id
FROM employees
WHERE LEVEL = 2
START WITH employee_id = (SELECT employee_id FROM employees WHERE
UPPER(last_name)
LIKE 'DE HAAN')
CONNECT BY PRIOR employee_id = manager_id;
```

--b)

```
SELECT LEVEL, employee_id, last_name, hire_date, salary, manager_id
FROM employees
START WITH employee_id = (SELECT employee_id FROM employees WHERE
UPPER(last_name)
LIKE 'DE HAAN')
CONNECT BY PRIOR employee_id = manager_id;
```

--Exercitiul 15--

```
SELECT LEVEL, employee_id, last_name, hire_date, salary, manager_id
FROM employees
START WITH employee_id =114
CONNECT BY PRIOR employee_id = manager_id;
```

```
SELECT LEVEL, employee_id, last_name, hire_date, salary, manager_id
FROM employees
START WITH employee_id =100
CONNECT BY PRIOR employee_id = manager_id;
```

--tema afisare last_name in functie de ierarhie

--tema 16,17,18

--EX16

```
SELECT level, employee_id, manager_id, last_name
FROM employees
WHERE level=3
```

```
START WITH LOWER(last_name) LIKE 'de haan'
CONNECT BY PRIOR employee_id=manager_id;
```

--EX17

```
SELECT level, employee_id, manager_id, LPAD(last_name, 3*level)
FROM employees
CONNECT BY PRIOR employee_id=manager_id;
```

--EX18, cu salary>15000

```
SELECT level, employee_id, last_name, salary, manager_id
FROM employees
--WHERE salary>15000
START WITH employee_id=(SELECT employee_id
                        FROM employees
                        WHERE salary=(SELECT MAX(salary)
                                      FROM employees
                                      )
                        )
CONNECT BY PRIOR salary>15000 AND PRIOR employee_id=manager_id;
```

-- 8 aprilie--

```
WITH
aux AS ( SELECT...)
aux2 AS (SELECT...)
...
auxn AS (SELECT...)
```

```
SELECT ...
FROM aux CROSS JOIN aux2 CROSS JOIN auxn
WHERE ...
...;
```

--ex 19--

```
WITH
total AS (SELECT department_id, SUM (salary) AS suma
          FROM employees
          GROUP BY department_id)
SELECT department_name, suma
FROM departments D
```

```

JOIN TOTAL T ON (D.department_id=T.department_id)
WHERE suma>(SELECT AVG(suma)
FROM TOTAL);

```

--20--

```

WITH
king AS (SELECT employee_id KOD
FROM employees
WHERE LOWER(last_name) LIKE 'king' AND LOWER(first_name) like 'steven')
SELECT employees.employee_id, first_name||' '||last_name, job_id, hire_date
FROM employees CROSS JOIN king
WHERE level=2 /*AND hire_date=(SELECT min(hire_date)
FROM employees)*/ AND EXTRACT(year FROM hire_date)!=1970
START WITH employees.employee_id=KOD
CONNECT BY PRIOR employees.employee_id=manager_id;

```

--21--

```

SELECT *
FROM ( SELECT last_name, salary
FROM employees
ORDER BY salary DESC)
WHERE ROWNUM<=10;

```

--22--

```

SELECT *
FROM (SELECT job_title
FROM jobs
ORDER BY (min_salary+max_salary)/2 ASC)
WHERE ROWNUM<=3;

```

--23--

```

SELECT 'departamentul '||department_name ||' este condus de '|| NVL(to_char(manager_id),
'nimeni')||' si '||
CASE WHEN nr>0 THEN 'are numarul de salariatii '||nr
ELSE 'nu are salariatii' END AS "informatii"
FROM departments D LEFT JOIN (SELECT department_id, COUNT(employee_id) NR
FROM employees
GROUP BY department_id) aux
ON (D.department_id=aux.department_id);

```

-- de testat daca merge cu decode--

--24--

```
SELECT last_name, first_name, length(last_name)
FROM employees
WHERE NULLIF(length(last_name), length(first_name)) IS NOT NULL;
```

--25--

```
SELECT last_name, hire_date, salary,
DECODE (to_char(hire_date, 'yyyy'), 1989, salary*1.2, 1990, salary*1.15, 1991, salary*1.10,
salary) marire
FROM employees;
```

--inlocuim decode cu case--

--26--

```
SELECT (SELECT sum(salary)
        FROM employees
        WHERE job_id like 'S%') suma, (SELECT AVG(salary)
        FROM employees
        WHERE (job_id, salary) IN (SELECT job_id, max(salary)
        FROM employees
        GROUP BY job_id)) medie
--, (SELECT min(salary) FROM employees WHERE job_id NOT LIKE 'S%' AND (job_id,
salary) != (SELECT ... CEL LUUUNG) minim FROM employees
FROM DUAL
```

--Cum s-ar face cu CASE?--

22.04.2020

LAB 6

```
SELECT * from works_on;
```

```
SELECT * from projects;
```

--EX 1

```
SELECT employee_id, last_name, salary
FROM employees e
WHERE NOT EXISTS ( SELECT 1
                   FROM projects p
```

```

WHERE EXTRACT(year FROM start_date) = 2006 AND
      EXTRACT(month FROM start_date) BETWEEN 1 AND 6
AND NOT EXISTS (SELECT 'q'
                 FROM works_on w
                 WHERE w.project_id = p.project_id
                 AND w.employee_id = e.employee_id
                 ));

```

--metoda 2

```

SELECT employee_id, last_name, salary
FROM employees JOIN works_on USING (employee_id)
WHERE project_id IN (SELECT project_id
                     FROM projects
                     WHERE EXTRACT(year FROM start_date) = 2006 AND
                           EXTRACT(month FROM start_date) BETWEEN 1 AND 6
                     )
GROUP BY employee_id, last_name, salary
HAVING COUNT(project_id) = (SELECT COUNT(project_id)
                             FROM projects
                             WHERE EXTRACT(year FROM start_date) = 2006 AND
                                   EXTRACT(month FROM start_date) BETWEEN 1 AND 6);

```

--Tema : metodele 3,4

--EX. 2

```

SELECT *
FROM projects p
WHERE NOT EXISTS (SELECT 2
                  FROM job_history jh
                  WHERE NOT EXISTS (SELECT 'd'
                                    FROM works_on w
                                    WHERE w.employee_id = jh.employee_id
                                    AND w.project_id = p.project_id)
                  GROUP BY employee_id
                  HAVING COUNT(job_id) = 2);

```

```

SELECT *
FROM job_history
ORDER BY 1;

```

--EX. 3

```

SELECT COUNT(COUNT(employee_id))
FROM job_history
GROUP BY employee_id
HAVING COUNT(job_id) >= 2;

```

--EX. 4


```
SELECT country_name, COUNT (*)
FROM employees
JOIN departments USING (department_id)
JOIN locations USING (location_id)
JOIN countries USING (country_id)
GROUP BY country_name;
```

--EX. 5

```
SELECT employee_id, last_name
FROM employees e
WHERE (SELECT COUNT(COUNT (w.project_id))
       FROM works_on w JOIN projects p ON (w.project_id = p.project_id)
       WHERE w.employee_id = e.employee_id AND delivery_date > deadline
       GROUP BY w.project_id ) >= 2;
```

--EX. 6

```
SELECT last_name, w.employee_id, w.project_id, p.project_name
FROM employees e
LEFT JOIN works_on w ON (e.employee_id = w.employee_id)
LEFT JOIN projects p ON (p.project_id = w.project_id);
```

--EX. 7

```
SELECT employee_id, last_name, salary
FROM employees
WHERE department_id IN (SELECT department_id
                       FROM employees
                       WHERE employee_id IN (SELECT project_manager
                                             FROM projects));
```

--EX. 8

```
SELECT employee_id, last_name, salary
FROM employees
MINUS
SELECT employee_id, last_name, salary
FROM employees
WHERE department_id IN (SELECT department_id
                       FROM employees
                       WHERE employee_id IN (SELECT project_manager
                                             FROM projects));
```

--EX. 9

```
SELECT department_id
FROM employees
GROUP BY department_id
HAVING AVG(salary) > &&p;
```

```
SELECT employee_id
FROM employees
WHERE salary = &w;
```

```
UNDEFINE p
```

```
ACCEPT w PROMPT 'w='
```

```
--EX. 10
```

```
SELECT last_name, first_name, salary, nr_proiecte
FROM employees e
JOIN (SELECT project_manager, COUNT(project_id) nr_proiecte
      FROM projects
      GROUP BY project_manager) t
ON (e.employee_id = t.project_manager)
WHERE nr_proiecte = 2;
```

```
--EX. 11
```

```
SELECT *
FROM employees e
WHERE NOT EXISTS (SELECT 'unu'
                  FROM projects p
                  WHERE project_manager = 102 AND
                  NOT EXISTS (SELECT 'altul'
                              FROM works_on w
                              WHERE w.employee_id = e.employee_id
                              AND w.project_id = p.project_id));
```

```
--Tema:12,13,14
```

```
--lab 6
```

```
-- ex 11
```

```
SELECT *
FROM works_on
ORDER BY 1,2;
```

```
SELECT *
FROM projects;
```

```
SELECT last_name, employee_id
FROM employees e
WHERE NOT EXISTS(SELECT '*'
                  FROM projects p
                  WHERE project_manager = 102 AND
```

```
-- ex 12
```

- b)
- varianta 4

```
-- ex 13
```

```
SELECT last_name, employee_id
FROM employees e
WHERE e.employee_id <> 200 AND NOT EXISTS(SELECT 1
      FROM works_on w
      WHERE employee_id = 200
      AND NOT EXISTS(SELECT 1
            FROM works_on
```

```

WHERE employee_id = e.employee_id AND project_id =
w.project_id
    )
    )
AND NOT EXISTS(SELECT project_id
FROM works_on w
WHERE e.employee_id = employee_id

MINUS

SELECT project_id
FROM works_on w1
WHERE employee_id = 200
);

```

-- ex 14

desc job_grades

```

select *
FROM job_grades;

```

```

SELECT last_name, employee_id, salary, grade_level
FROM employees
CROSS JOIN job_grades
WHERE salary BETWEEN LOWEST_SAL AND highest_sal;

```

-- ex 15

```

SELECT employee_id, last_name, salary, department_id
FROM employees WHERE employee_id = &p_cod;

```

```

DEFINE p_cod -- Ce efect are?
SELECT employee_id, last_name, salary, department_id
FROM employees WHERE employee_id = &p_cod;
UNDEFINE p_cod

```

```

DEFINE p_cod = 100
SELECT employee_id, last_name, salary, department_id
FROM employees WHERE employee_id = &p_cod;
UNDEFINE p_cod

```

```

ACCEPT p_cod PROMPT "cod= "
SELECT employee_id, last_name, salary, department_id
FROM employees WHERE employee_id = &p_cod;

```

-- ex 16

```
ACCEPT p_jobld PROMPT "job_id= "  
SELECT employee_id, department_id, last_name, salary * 12  
FROM employees  
WHERE UPPER(job_id) = '&p_jobld';
```

-- ex 17

```
ACCEPT p_date PROMPT "data ="  
SELECT employee_id, department_id, last_name, salary * 12  
FROM employees  
WHERE HIRE_DATE >= TO_DATE('&p_date', 'DD-MM-YYYY');  
UNDEFINE p_date
```

-- ex 18

```
SELECT *  
FROM (SELECT &p_coloana  
      FROM &p_tabel  
      ORDER BY &p_coloana  
      )  
WHERE ROWNUM <= 5;
```

-- ex 19

```
ACCEPT p_date1 PROMPT "data1 ="  
ACCEPT p_date2 PROMPT "data2 ="  
SELECT last_name || ', ' || job_id "Angajati", hire_date  
FROM employees  
WHERE HIRE_DATE BETWEEN TO_DATE('&p_date1', 'MM/DD/YY') AND  
TO_DATE('&p_date2', 'MM/DD/YY');
```

-- ex 20 TEMA! (folositi CITY)

-- ex 21

```
UNDEFINE p_date1  
UNDEFINE p_date2
```

```
ACCEPT p_date1 PROMPT "data1 ="  
ACCEPT p_date2 PROMPT "data2 ="  
SELECT TO_DATE('&p_date1', 'MM/DD/YY') + ROWNUM -1  
FROM DUAL  
CONNECT BY ROWNUM < TO_DATE('&p_date2', 'MM/DD/YY') - TO_DATE('&p_date1',  
'MM/DD/YY') + 1;
```

-- b) TEMA!!

-- LABORATOR 7

```
CREATE TABLE emp_ama AS SELECT * FROM EMPLOYEES;  
CREATE TABLE dept_ama AS SELECT * FROM departments;
```

```
desc emp_ama  
desc employees  
desc dept_ama
```

```
SELECT *  
FROM user_constraints  
WHERE UPPER(table_name) LIKE ('EMP_%')  
ORDER BY table_name;
```

-- ex 3 datele sunt identice, structura tabelelor nu e la fel

```
SELECT *  
FROM emp_ama;
```

-- ex 4

```
ALTER TABLE emp_ama  
ADD CONSTRAINT pk_emp_ama PRIMARY KEY(employee_id);
```

```
ALTER TABLE dept_ama  
ADD CONSTRAINT pk_dept_ama PRIMARY KEY(department_id);
```

```
ALTER TABLE emp_ama  
ADD CONSTRAINT fk_emp_ama FOREIGN KEY(department_id) REFERENCES  
dept_ama(department_id);
```

-- ex 5

```
INSERT INTO DEPT_ama -- eroare not enough values  
VALUES (300, 'Programare');
```

```
INSERT INTO DEPT_ama (department_id, department_name)  
VALUES (300, 'Programare');
```

```
INSERT INTO DEPT_ama (department_name, department_id) -- eroare semantica diferita  
VALUES (300, 'Programare');
```

```
INSERT INTO DEPT_ama (department_id, department_name, location_id) -- unique  
constraint (GRUPA44.PK_DEPT_AMA) violated  
VALUES (300, 'Programare', null); -- inserez din nou cu aceeași primary key
```

```
INSERT INTO DEPT_ama (department_name, location_id) -- cannot insert NULL into
primary key field
VALUES ('Programare', null);
```

-- ex 6

```
select *
FROM dept_ama;
```

```
INSERT INTO emp_ama(employee_id, last_name, department_id, job_id, hire_date, email)
VALUES (EMPLOYEES_SEQ.nextval, 'Manolache', 300, 'IT_PROG', SYSDATE,
'andrei@yahoo.com');
```

```
select *
FROM emp_ama;
```

```
COMMIT;
```

-- tema pana la 10!

LAB 7

--ex 6--

```
SELECT * FROM emp_rdu;
INSERT INTO EMP_RDU VALUES (EMPLOYEES_SEQ.nextval, NULL, 'ceva',
'ana@palmier', NULL, SYSDATE, 'jobul', NULL, NULL, NULL, NULL);
COMMIT;
```

--ex 7--

```
SELECT * FROM DEPT_RDU;
INSERT INTO dept_rdu (department_id, department_name) VALUES (300, 'Programare');
INSERT INTO EMP_RDU (employee_id, last_name, hire_date, job_id, email,
department_id) VALUES (EMPLOYEES_SEQ.nextval, 'altceva', SYSDATE, 'alt job',
'mirela@ionela', 300);
COMMIT;
```

--ex 8--

```
INSERT INTO EMP_RDU (employee_id, last_name, hire_date, job_id, email)
VALUES ((SELECT MAX(EMPLOYEE_ID)+1 FROM emp_rdu), 'inna', SYSDATE,
'celalatjob', 'mini@email');
```

--varianta 2--

```
INSERT INTO EMP_RDU (employee_id, last_name, hire_date, job_id, email)
SELECT (SELECT MAX(EMPLOYEE_ID) +1 FROM EMP_RDU), 'inna', SYSDATE,
'celalatjob', 'mini@email'
FROM DUAL;
```

--ex 9--

```
CREATE TABLE emp1_abc as SELECT * FROM EMPLOYEES WHERE 1=-1;
INSERT INTO emp1_abc
SELECT * FROM employees WHERE commission_pct > 0.25;
```

--ex 10--

```
INSERT INTO EMP_RDU (employee_id, last_name, first_name, hire_date, job_id, email,
salary, commission_pct)
VALUES (0, USER, USER, sysdate, 'TOTAL', 'TOTAL', (SELECT sum(salary) FROM
emp_rdu), (SELECT sum (commission_pct)/ count(*) FROM emp_rdu));
```

--ex 11--

```
INSERT INTO EMP_RDU (employee_id, last_name, first_name, hire_date, job_id, email,
salary)
VALUES (&p_cod, '&&p_num', '&&p_prenume', sysdate, 'oarecare', substr('&p_prenume',
1, 1) || substr('&p_num', 1, 7), &p_salariu);
--undefine pentru next insertion--
```

--ex 12--

```
CREATE TABLE emp2_abc as SELECT * FROM EMPLOYEES WHERE 1=-1;
CREATE TABLE emp3_abc as SELECT * FROM EMPLOYEES WHERE 1=-1;
```

```
INSERT ALL
WHEN salary < 5000 THEN INTO emp1_abc
WHEN salary between 5000 and 10000 THEN INTO emp2_abc
ELSE INTO emp3_abc
SELECT * FROM employees;
```

--ex 13 tema--

--ex 14--


```
UPDATE emp_rdu
SET salary=salary*1.05;
ROLLBACK;
```

--ex 15--

```
UPDATE emp_rdu
SET job_id='SA_REP'
WHERE department_id=80;
```

--ex 16--

```
UPDATE dept_rdu
SET manager_id=(SELECT employee_id FROM emp_rdu WHERE lower(first_name)||'
'||lower(last_name)='douglas grant')
WHERE department_id=20;
```

```
UPDATE emp_rdu
SET salary=salary+1000
WHERE employee_id=(SELECT employee_id FROM emp_rdu WHERE lower(first_name)||'
'||lower(last_name)='douglas grant');
```

--ex 17--

```
UPDATE emp_rdu sub
SET (salary, commission_pct) = (SELECT salary, commission_pct FROM employees
WHERE sub.manager_id=employee_id)
WHERE salary=(SELECT MIN(salary) FROM employees);
```

--tema pana la 21--

--Lab 7 - 13 mai 2020 --

-- Ex 18 --

```
UPDATE emp_ado
```

```
SET email = INITCAP(last_name)||NVL(first_name, '.')
```

```
WHERE (department_id, salary) IN (SELECT department_id, MAX(salary)
```

```
FROM employees GROUP BY department_id);
```

```
ROLLBACK;
```

```
-- Ex 22 --
```

```
SELECT * from dept_ado;
```

```
DELETE FROM dept_ado;
```

```
-- Daca aveam constrangere de cheie straina, nu s-ar fi sters in cazul in care angajatii lucrau  
in acel departament
```

```
-- Ex 23 --
```

```
DELETE FROM emp_ado
```

```
WHERE commission_pct is NULL;
```

```
ROLLBACK;
```

```
-- EX 24 --
```

```
DELETE FROM dept_rdu d
```

```
WHERE NOT EXISTS (SELECT 1 FROM employees
```

```
WHERE d.department_id = department_id );
```

```
ROLLBACK;
```

```
SELECT * FROM user_constraints WHERE table_name LIKE 'EMP%' ORDER BY 4;
```

```
-- SAU --
```

```
CREATE TABLE dept_2 AS SELECT * FROM departments;
```

```
CREATE TABLE emp_2 AS SELECT * FROM employees;
```

```
DELETE FROM dept_2
```

```
WHERE department_id NOT IN (SELECT NVL(department_id,0) FROM employees );
```

```
SELECT * FROM dept_2;
```

```
ROLLBACK;
```

```
DELETE FROM dept_2 d
WHERE NOT EXISTS (SELECT 1 FROM employees
                  WHERE d.department_id = department_id );

ROLLBACK;

-- EX 25 --

ACCEPT p_cod PROMPT "Dati un cod de angajat:";

SELECT * FROM emp_ado
WHERE employee_id = &p_cod;

DELETE FROM emp_ado
WHERE employee_id = &p_cod;

SAVEPOINT A;

-- EX 28 --

DELETE FROM emp_ado;

SELECT * FROM emp_ado;

ROLLBACK TO A;

-- COMANDA MERGE --

MERGE INTO emp_ado x
USING employees e
ON (x.employee_id = e.employee_id)

WHEN MATCHED THEN

    UPDATE SET

        x.first_name = e.first_name,

        x.last_name = e.last_name,

        x.email = e.email,

        x.phone_number = e.phone_number,
```

```
x.hire_date = e.hire_date,  
x.job_id = e.job_id,  
x.salary = e.salary,  
x.commission_pct = e.commission_pct,  
x.manager_id = e.manager_id,  
x.department_id = e.department_id
```

WHEN NOT MATCHED THEN

```
INSERT VALUES (e.employee_id, e.first_name, e.last_name, e.email,  
e.phone_number,e.hire_date,  
e.job_id, e.salary, e.commission_pct, e.manager_id, e.department_id);
```

```
SELECT * FROM emp_ado;
```

```
-- LABORATORUL 8 --
```

```
-- EX 1 --
```

```
-- A) --
```

```
CREATE TABLE ANGAJATI_ADO (
```

```
cod_ang NUMBER(4),  
nume VARCHAR2(20),  
prenume VARCHAR2(20),  
email CHAR(15),  
data_ang DATE,  
job VARCHAR2(10),  
cod_sef NUMBER(4),  
salariu NUMBER(8,2),  
cod_dep NUMBER(2));
```

DESC ANGAJATI_ADO

DROP TABLE ANGAJATI_ADO;

-- B) --

CREATE TABLE ANGAJATI_ADO (

cod_ang NUMBER(4) PRIMARY KEY,

nume VARCHAR2(20) NOT NULL,

prenume VARCHAR2(20),

email CHAR(15),

data_ang DATE,

job VARCHAR2(10),

cod_sef NUMBER(4),

salariu NUMBER(8,2) NOT NULL,

cod_dep NUMBER(2));

-- C) --

DROP TABLE ANGAJATI_ADO;

CREATE TABLE ANGAJATI_ADO (

cod_ang NUMBER(4),

nume VARCHAR2(20) NOT NULL,

prenume VARCHAR2(20),

email CHAR(15),

data_ang DATE,

job VARCHAR2(10),

```
cod_sef NUMBER(4),  
salariu NUMBER(8,2) NOT NULL,  
cod_dep NUMBER(2),  
CONSTRAINT ang_pk PRIMARY KEY (cod_ang)  
);
```

```
DROP TABLE ANGAJATI_ADO;
```

```
CREATE TABLE ANGAJATI_ADO (  
    cod_ang NUMBER(4),  
    nume VARCHAR2(20) NOT NULL,  
    prenume VARCHAR2(20),  
    email CHAR(15),  
    data_ang DATE default SYSDATE,  
    job VARCHAR2(10),  
    cod_sef NUMBER(4),  
    salariu NUMBER(8,2) NOT NULL,  
    cod_dep NUMBER(2),  
    CONSTRAINT ang_pk PRIMARY KEY (cod_ang)  
);
```

```
-- EX 2 --
```

```
INSERT INTO ANGAJATI_ADO (cod_ang, nume, prenume, data_ang, job, salariu,  
cod_dep)
```

```
VALUES (100, 'Nume1', 'Prenume1', null, 'Director', 20000, 10);
```

```
INSERT INTO ANGAJATI_ADO (cod_ang, nume, prenume, email, data_ang, job,cod_sef, salariu, cod_dep)
```

```
VALUES (101, 'Nume2', 'Prenume2','Nume2', TO_DATE('02-02-2004', 'DD-MM-YYYY'), 'Inginer',100, 10000, 10);
```

```
INSERT INTO ANGAJATI_ADO (cod_ang, nume, prenume, email, data_ang, job,cod_sef, salariu, cod_dep)
```

```
VALUES (102, 'Nume3', 'Prenume3','Nume3', TO_DATE('05-06-2000', 'DD-MM-YYYY'), 'Analist',101, 5000, 20);
```

```
INSERT INTO ANGAJATI_ADO (cod_ang, nume, prenume, data_ang, job,cod_sef, salariu, cod_dep)
```

```
VALUES (103, 'Nume4', 'Prenume4', null, 'Inginer',100, 9000, 20);
```

```
INSERT INTO ANGAJATI_ADO (cod_ang, nume, prenume,email, data_ang, job,cod_sef, salariu, cod_dep)
```

```
VALUES (104, 'Nume5', 'Prenume5', 'Nume5', null, 'Analist',101, 3000, 30);
```

```
-- Ex 3 --
```

```
CREATE TABLE ANGAJATI10_ado AS
```

```
SELECT * FROM ANGAJATI_ADO
```

```
WHERE cod_dep = 10;
```

```
SELECT * FROM ANGAJATI10_ado;
```

```
DESC ANGAJATI_ADO
```

```
DESC ANGAJATI10_ado
```

```
SELECT * FROM USER_CONSTRAINTS WHERE TABLE_NAME LIKE 'ANGAJATI%';
```

```
-- EX 4 --
```

```
ALTER TABLE ANGAJATI_ADO
```

```
ADD comision NUMBER(4,2);
```

```
desc ANGAJATI_ADO
```

```
-- EX 5 --
```

```
ALTER TABLE ANGAJATI_ADO
```

```
MODIFY salariu NUMBER(6,2); -- NU SE POATE
```

```
ALTER TABLE ANGAJATI_ADO
```

```
MODIFY salariu NUMBER(9,2);
```

```
desc ANGAJATI_ADO
```

```
-- EX 6 --
```

```
ALTER TABLE ANGAJATI_ADO
```

```
MODIFY salariu DEFAULT 0;
```

```
SELECT * FROM ANGAJATI_ADO;
```

```
-- EX 7 --
```

```
ALTER TABLE ANGAJATI_ADO
```

```
MODIFY ( comision NUMBER(2,2), salariu NUMBER(10,2));
```

```
desc ANGAJATI_ADO
```

```
-- EX 8 --
```

```
UPDATE ANGAJATI_ADO
```

```
SET comision = 0.1
```

```
WHERE job LIKE 'A%';
```

```
-- EX 9 --
```

```
ALTER TABLE ANGAJATI_ADO
```

```
MODIFY email VARCHAR2; -- nu ruleaza pt ca nu avem dimensiune
```

```
-- EX 10 --
```

```
ALTER TABLE ANGAJATI_ADO
```

```
ADD nr_telefon NUMBER DEFAULT 0;
```

```
-- EX 11 --
```

```
SELECT * FROM ANGAJATI_ADO;
```

```
ALTER TABLE ANGAJATI_ADO
```



```
DROP COLUMN nr_telefon;
```

```
ROLLBACK; -- NU REVIN MODIFICARILE
```

```
-- EX 12 --
```

```
RENAME ANGAJATI_ADO TO ANGAJATI3_ADO;
```

```
-- EX 13 --
```

```
RENAME ANGAJATI3_ADO TO ANGAJATI_ADO;
```

```
SELECT * FROM TAB;
```

```
-- EX 14 --
```

```
TRUNCATE TABLE ANGAJATI_ADO;
```

```
SELECT * FROM ANGAJATI_ADO;
```

```
ROLLBACK;
```

```
-- EX 15 --
```

```
CREATE TABLE DEPARTAMENTE_ADO (
```

```
    cod_dep NUMBER(2),
```

```
    nume VARCHAR2(15),
```

```
    cod_director NUMBER(4));
```

```
DROP TABLE DEPARTAMENTE_ADO;
```

```
CREATE TABLE DEPARTAMENTE_ADO (
```

```
    cod_dep NUMBER(2),
```

```
    nume VARCHAR2(15) NOT NULL,
```

```
    cod_director NUMBER(4));
```

```
INSERT INTO DEPARTAMENTE_ADO
```

```
VALUES (10, 'Administrativ', 100);
```

```
INSERT INTO DEPARTAMENTE_ADO
```

```
VALUES (20, 'Proiectare', 101);
```

```
INSERT INTO DEPARTAMENTE_ADO
```

```
VALUES (30, 'Programare', null);
```

```
-- EX 17 --
```

```
ALTER TABLE DEPARTAMENTE_ADO
```

```
ADD CONSTRAINT dep_pk_ado PRIMARY KEY (cod_dep);
```

```
-- EX 18 --
```

```
-- A) --
```

```
ALTER TABLE ANGAJATI_ADO
```

```
ADD CONSTRAINT emp_fk_ado FOREIGN KEY (cod_dep)
```

```
REFERENCES DEPARTAMENTE_ADO (cod_dep);
```

```
-- B) --
```

```
DROP TABLE ANGAJATI_ADO;
```

```
CREATE TABLE ANGAJATI_ADO (
```

```
    cod_ang NUMBER(4) PRIMARY KEY,
```

```
    nume VARCHAR2(20) NOT NULL,
```

```
    prenume VARCHAR2(20),
```

```
    email CHAR(15) UNIQUE,
```

```
    data_ang DATE default SYSDATE,
```

```
    job VARCHAR2(10),
```

```
    cod_sef NUMBER(4) REFERENCES ANGAJATI_ADO (cod_ang),
```

```
    salariu NUMBER(8,2) NOT NULL,  
  
    cod_dep NUMBER(2) CHECK (cod_dep > 0) REFERENCES  
DEPARTAMENTE_ADO (cod_dep),  
  
    comision NUMBER(2,2),  
  
    CONSTRAINT ang_ado_u UNIQUE(nume, prenume),  
  
    CONSTRAINT ang2_ado_ck CHECK (salariu > comision*100)  
  
);
```

-- laborator 8 - 20 mai 2020 -

-- ex 19 --

```
DROP TABLE ANGAJATI_ADO;
```

```
CREATE TABLE ANGAJATI_ADO (  
    cod_ang NUMBER(4),  
    nume VARCHAR2(20) NOT NULL,  
    prenume VARCHAR2(20),  
    email CHAR(15),  
    data_ang DATE default SYSDATE,  
    job VARCHAR2(10),  
    cod_sef NUMBER(4),  
    salariu NUMBER(8,2) NOT NULL,  
    cod_dep NUMBER(2),  
    comision NUMBER(2,2),  
    CONSTRAINT ang_ado_u UNIQUE(nume, prenume),
```

```

        CONSTRAINT ang2_ado_ck CHECK (salariu > comision*100),

        CONSTRAINT ang_ado_pk PRIMARY KEY (cod_ang),

        CONSTRAINT ang_ado_fk1 FOREIGN KEY (cod_sef) REFERENCES
ANGAJATI_ADO (cod_ang),

        CONSTRAINT ang_ado_check2 CHECK (cod_dep > 0), FOREIGN KEY (cod_dep)
REFERENCES DEPARTAMENTE_ADO (cod_dep),

        CONSTRAINT ang_ado_u2 UNIQUE(email)

    );

```

```

SELECT * FROM USER_CONSTRAINTS WHERE LOWER(table_name) LIKE
'%angajati%' ORDER BY 4;

```

```

-- ex 20 --

```

```

INSERT INTO ANGAJATI_ADO (cod_ang, nume, prenume, data_ang, job, salariu,
cod_dep)

```

```

VALUES (100, 'Nume1', 'Prenume1', null, 'Director', 20000, 10);

```

```

INSERT INTO ANGAJATI_ADO (cod_ang, nume, prenume, email, data_ang, job,cod_sef,
salariu, cod_dep)

```

```

VALUES (101, 'Nume2', 'Prenume2','Nume2', TO_DATE('02-02-2004', 'DD-MM-YYYY'),
'Inginer',100, 10000, 10);

```

```

INSERT INTO ANGAJATI_ADO (cod_ang, nume, prenume, email, data_ang, job,cod_sef,
salariu, cod_dep)

```

```

VALUES (102, 'Nume3', 'Prenume3','Nume3', TO_DATE('05-06-2000', 'DD-MM-YYYY'),
'Analist',101, 5000, 20);

```

```

INSERT INTO ANGAJATI_ADO (cod_ang, nume, prenume, data_ang, job,cod_sef, salariu,
cod_dep)

```

```

VALUES (103, 'Nume4', 'Prenume4', null, 'Inginer',100, 9000, 20);

```

```

INSERT INTO ANGAJATI_ADO (cod_ang, nume, prenume,email, data_ang, job,cod_sef,
salariu, cod_dep)

```

```
VALUES (104, 'Nume5', 'Prenume5', 'Nume5', null, 'Analist', 101, 3000, 30);
```

```
SELECT * FROM ANGAJATI_ADO;
```

```
-- EX 21 --
```

```
DROP TABLE DEPARTAMENTE_ADO; -- nu putem suprima din cauza constrangerii de  
cheie straina pe cod_dep
```

```
SELECT * FROM DEPARTAMENTE_ADO;
```

```
-- EX 22 --
```

```
SELECT * FROM TAB;
```

```
SELECT * FROM USER_TABLES;
```

```
-- EX 23 a)--
```

```
SELECT * FROM USER_CONSTRAINTS WHERE LOWER(table_name) IN ('angajati_ado',  
'departamente_ado') ORDER BY 4;
```

```
-- b)
```

```
SELECT table_name, constraint_name, column_name FROM USER_CONS_COLUMNS  
WHERE LOWER(table_name) IN ('angajati_ado', 'departamente_ado');
```

-- EX 24 --

```
ALTER TABLE ANGAJATI_ADO MODIFY email NOT NULL;
```

```
UPDATE ANGAJATI_ADO
```

```
    SET email = 'MAIL2' WHERE cod_ang = 100;
```

```
UPDATE ANGAJATI_ADO
```

```
    SET email = 'MAIL1' WHERE cod_ang = 103;
```

```
SELECT * FROM ANGAJATI_ADO;
```

-- EX 25 --

```
SELECT * FROM DEPARTAMENTE_ADO;
```

```
INSERT INTO ANGAJATI_ADO (cod_ang, nume, prenume,email, data_ang, job,cod_sef,  
salariu, cod_dep)
```

```
VALUES (105, 'Nume6', 'Prenume6', 'Nume6', null, 'Analist',101, 3000, 50);
```

```
-- NU SE POATE DEOARECE EXISTA CONSTRANGERE DE CHEIE STRAINA PE  
COD_DEP SI NU AVEM DEP 50 IN TABEL
```

```
SELECT * FROM USER_CONSTRAINTS WHERE LOWER(table_name) IN ('angajati_ado',  
'departamente_ado') ORDER BY 4;
```

-- EX 26 --

```
INSERT INTO DEPARTAMENTE_ADO VALUES (60, 'Analiza', null);
```

```
COMMIT;
```

```
-- EX 27 --
```

```
DELETE FROM DEPARTAMENTE_ADO WHERE cod_dep = '20';
```

```
-- NU SE POATE STERGE PT CA EXISTA ANGAJATI DIN ANGAJATI_ADO IN DEP 20
```

```
-- EX 28 --
```

```
DELETE FROM DEPARTAMENTE_ADO WHERE cod_dep = '60'; -- MERGE DEOARECE  
NU EXISTA ANGAJATI IN DEP 60
```

```
ROLLBACK;
```

```
-- EX 29 --
```

```
INSERT INTO ANGAJATI_ADO (cod_ang, nume, prenume,email, data_ang, job,cod_sef,  
salariu, cod_dep)
```

```
VALUES (106, 'Nume7', 'Prenume7', 'Nume7', null, 'Analist',114, 3000, 20);
```

```
-- NU MERGE DEOARECE ANG CU CODUL 114 NU EXISTA
```

```
-- EX 30 --
```

```
INSERT INTO ANGAJATI_ADO (cod_ang, nume, prenume,email, data_ang, job,cod_sef,  
salariu, cod_dep)
```

```
VALUES (114, 'Nume7', 'Prenume7', 'Nume7', null, 'Analist',null, 3000, 20);
```

```
INSERT INTO ANGAJATI_ADO (cod_ang, nume, prenume,email, data_ang, job,cod_sef, salariu, cod_dep)
```

```
VALUES (106, 'Nume8', 'Prenume8', 'Nume8', null, 'Analist',114, 3000, 20);
```

```
-- CONCLUZIE --
```

```
-- INSERAM LINII IN TABELUL PARINTE SI APOI IN TABELUL COPIL
```

```
-- EX 31 --
```

```
ALTER TABLE ANGAJATI_ADO
```

```
    DROP CONSTRAINT SYS_C00348894;
```

```
ALTER TABLE ANGAJATI_ADO
```

```
ADD CONSTRAINT ANG_ADO_FK2 FOREIGN KEY (cod_dep) REFERENCES  
DEPARTAMENTE_ADO (cod_dep)ON DELETE CASCADE;
```

```
-- EX 32 --
```

```
DELETE FROM DEPARTAMENTE_ADO WHERE cod_dep = 20;
```

```
SELECT * FROM ANGAJATI_ADO;
```

```
SELECT * FROM DEPARTAMENTE_ADO;
```

```
ROLLBACK;
```


-- EX 33 --

ALTER TABLE DEPARTAMENTE_ADO

ADD CONSTRAINT dep_ado_fk FOREIGN KEY (cod_DIRECTOR) REFERENCES
ANGAJATI_ADO(cod_ang) ON DELETE SET NULL;

-- EX 34 --

SELECT * FROM DEPARTAMENTE_ADO;

UPDATE DEPARTAMENTE_ADO

SET cod_director = 102 WHERE cod_dep = 30;

DELETE FROM ANGAJATI_ADO WHERE cod_ang = 102;

ROLLBACK;

DELETE FROM ANGAJATI_ADO WHERE cod_ang = 101;

-- EX 35 --

ALTER TABLE ANGAJATI_ADO

ADD CONSTRAINT ang_ado_check3 CHECK (salariu < 30000);

-- ex 36 --

```
UPDATE ANGAJATI_ADO
```

```
SET salariu = 35000 WHERE cod_ang = 100; -- depaseset 30000
```

```
-- ex 37
```

```
ALTER TABLE ANGAJATI_ADO
```

```
MODIFY CONSTRAINT ang_ado_check3 DISABLE;
```

```
UPDATE ANGAJATI_ADO
```

```
SET salariu = 35000 WHERE cod_ang = 100;
```

```
ALTER TABLE ANGAJATI_ADO
```

```
MODIFY CONSTRAINT ang_ado_check3 ENABLE; -- NU MERGE DEOARECE EXISTA  
UN SALARIU PESTE 30 000 IN TABEL
```