1. // lab 2 26 - 02 - 2020

Scrieţi 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, utilizaţi atât funcţia

CONCAT cât şi operatorul “||”.

/\* ex1 \*/

SELECT first\_name , last\_name || ' castiga ' || salary || ' lunar, dar doreste ' || salary \* 3 as " SALARIU IDEAL "

FROM EMPLOYEES;

/\* ex 2 \*/

Scrieţi o cerere prin care să se afişeze prenumele salariatului cu prima litera majusculă şi

toate celelalte litere mici, numele acestuia cu majuscule şi lungimea numelui, pentru

angajaţii 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 soluţii (cu operatorul LIKE şi funcţia 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 afişeze pentru angajaţii 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 \* FROM EMPLOYEES;

SELECT employee\_id , last\_name , department\_id

FROM employees

WHERE UPPER(TRIM(first\_name)) LIKE 'STEVEN';

/\* EX4 \*/

Să se afişeze pentru toţi angajaţii al căror nume se termină cu litera 'e', codul, numele,

lungimea numelui şi poziţia din nume în care apare prima data litera 'a'. Utilizaţi 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 afişeze detalii despre salariaţii 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 afişeze codul salariatului, numele, salariul, salariul mărit cu 15%, exprimat cu două

zecimale şi numărul de sute al salariului nou rotunjit la 2 zecimale. Etichetaţi ultimele două

coloane “Salariu nou”, respectiv “Numar sute”. Se vor lua în considerare salariaţii 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 şi data angajării salariaţilor care câştigă comision. Să se eticheteze coloanele „Nume angajat”, „Data angajarii”. Pentru a nu obţine alias-ul datei angajării trunchiat, utilizaţi funcţia 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 afişeze 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 afişeze numărul de zile rămase până la sfârşitul anului.

SELECT ROUND(TO\_DATE('31-DEC-2020', 'dd-mon-yyyy') - SYSDATE)

FROM DUAL;

/\* EX10 \*/

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

b) Să se afişeze 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 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. Etichetaţi 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 afişeze numele şi numărul de luni de la data angajării. Etichetaţi 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 afişeze numele, data angajării şi ziua săptămânii în care a început lucrul fiecare salariat. Etichetaţi coloana “Zi”. Ordonaţi 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 afişeze numele angajaţilor şi comisionul. Dacă un angajat nu câştigă comision, să

se scrie “Fara comision”. Etichetaţi 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 şi comisionul tuturor angajaţilor al căror venit lunar

depăşeşte 10000$.

SELECT last\_name, salary, commission\_pct   
FROM employees  
WHERE salary\*(1+NVL(commission\_pct,0))>10000;

\*/ex 16\*/

16. Sa 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 ceilalţi

angajaţi 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 renegogiat"

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 renegogiat"

FROM employees;

/\* ex17 \*/

17. Să se afişeze numele salariatului, codul şi numele departamentului pentru toţi angajaţii.

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.

--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 afişeze numele angajatului, numele departamentului şi locatia pentru toţi angajaţii

care câştigă 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 afişeze numele salariatului şi numele departamentului pentru toţi salariaţii 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 afişeze numele, job-ul, codul şi numele departamentului pentru toţi angajaţii 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 afişeze codul angajatului şi numele acestuia, împreună cu numele şi codul şefului

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 afişa toţi salariaţii, inclusiv cei care nu au şef.

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. Creaţi o cerere care să afişeze numele angajatului, codul departamentului şi toţi salariaţii

care lucrează în acelaşi 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 afişeze numele şi data angajării pentru salariaţii care au fost angajaţi 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 afişeze suma salariilor pe departamente şi, 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 afişeze 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 aceeaşi zi a lunii în care cei mai multi

dintre salariati 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

-- Exercitiul 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);

--Exercitiul 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 lower

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 salariati '||nr

ELSE 'nu are salariati' 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

NOT EXISTS (SELECT 1

FROM works\_on

WHERE employee\_id = e.employee\_id

AND project\_id = p.project\_id

)

);

-- ex 12

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

)

);

-- b)

-- varianta 4

SELECT DISTINCT e.last\_name, e.employee\_id

FROM employees e

JOIN works\_on aux ON(aux.employee\_id = e.employee\_id)

WHERE e.employee\_id <> 200 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 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\_jobId PROMPT "job\_id= "

SELECT employee\_id, department\_id, last\_name, salary \* 12

FROM employees

WHERE UPPER(job\_id) = '&p\_jobId';

-- 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 aceeasi 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\_nume', '&&p\_prenume', sysdate, 'oarecare', substr('&p\_prenume', 1, 1) || substr('&p\_nume', 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