Lab-1

1. Găsiți structura completă a tabelelor employees, departments, locations și jobs.
    a. DESCRIBE employees;
    DESCRIBE departments;
    DESCRIBE locations;
    DESCRIBE jobs;
2. Afișați conținutul tabelelor departments, employees, locations, jobs și job_history.
    a. SELECT * FROM departments;
    SELECT * FROM employees;
    SELECT * FROM locations;
    SELECT * FROM jobs;
    SELECT * FROM job_history;
3. Se dorește o listă cu numele, identificatorul jobului, data angajării și identificatorul
angajatului pentru fiecare persoană angajată. Numele câmpurilor afișate vor fi exact cele
menționate mai sus.
    a. SELECT first_name || ' ' || last_name AS nume, job_id AS identificator_job, hire_date AS data_angajarii, employee_id AS identificator_angajat
    FROM employees;
4. Listă cu numele angajaților concatenate cu identificatorii joburilor acestora, separată prin virgulă și spațiu
    a. SELECT first_name || ' ' || last_name || ', ' || job_id AS Angajat_functie
    FROM employees;
5. Listă cu numele și salariile angajaților care câștigă mai mult de 5000 pe lună
    a. SELECT first_name || ' ' || last_name AS nume, salary
    FROM employees
    WHERE salary > 5000;

6. Angajați care câștigă mai mult decât oricare din angajații departamentului 60
    a. SELECT first_name || ' ' || last_name AS nume, salary
    FROM employees
    WHERE salary > ANY (SELECT salary FROM employees WHERE department_id = 60);
7. Angajați care câștigă mai mult decât toți angajații departamentului 60
    a. SELECT first_name || ' ' || last_name AS nume, salary
    FROM employees
    WHERE salary > ALL (SELECT salary FROM employees WHERE department_id = 60);
8. Creați o listă cu angajații care ocupă, cel puțin pentru a doua oară, aceeași funcție.
    (puteți utiliza operatorul intersecție)
    a. SELECT employee_id, job_id
    FROM job_history
    INTERSECT
    SELECT employee_id, job_id
    FROM employees;
9. Nume, prenume, job și data angajării pentru Matos și Taylor, ordonat după vechime
    a. SELECT first_name, last_name, job_id, hire_date
    FROM employees
    WHERE last_name IN ('Matos', 'Taylor')
    ORDER BY hire_date DESC;
10. Afișați angajații care au în prenume, pe poziția a treia litera a.
    a. SELECT first_name, last_name
    FROM employees
    WHERE SUBSTR(first_name, 3, 1) = 'a';
11. Afișați data sistem. Etichetați coloana cu Data_curenta.
    a. SELECT SYSDATE AS Data_curenta FROM dual;

12. Se dorește o listă cu perioadele în care au lucrat angajații în companie. Pentru aceasta
    se va afișa numele fiecărui angajat și se va calcula numărul de luni întregi de la angajare.
    Se va eticheta această coloană cu Luni_lucrate.
    a. SELECT first_name || ' ' || last_name AS nume,
       FLOOR(MONTHS_BETWEEN(SYSDATE, hire_date)) AS Luni_lucrate
       FROM employees;
13. Scrieți o interogare care să afișeze numele angajaților și comisionul încasat. Dacă
    angajatul nu primește comision, se va scrie Fără comision.
    a. SELECT last_name, NVL(TO_CHAR(commission_pct),'fara comision') AS without_commision from employees;
14. Realizați o listă care să conțină numărul de angajați pentru fiecare job.
    a. SELECT job_id, COUNT(*) AS numar_angajati
       FROM employees
       GROUP BY job_id;
15. Găsiți numărul managerilor din companie (fără a afișa detalii despre aceștia).
    a. SELECT COUNT(*) AS Nr_manager
       FROM employees
       WHERE employee_id = ANY(SELECT manager_id FROM employees
       WHERE manager_id IS NOT NULL);
    b. SELECT COUNT(DISTINCT(manager_id)) from employees;
16. Afișați diferența dintre salariul maxim și cel minim încasat și etichetați coloana cu
    Diferente_salariale.
    a. SELECT MAX(salary) - MIN(salary) AS Diferente_salariale
       FROM employees;
17. Scrieți o interogare care să întoarcă numele și prenumele angajaților, numele departamentelor în care lucrează și orașul în care se găsesc.

a. SELECT e.last_name, e.first_name, d.department_name, l.city FROM employees e, departments d, locations l
WHERE e.department_id = d.department_id AND
d.location_id=l.location_id;

18. Scrieți o interogare care să afișeze numele și data angajării muncitorilor și ale managerilor cărora le sunt subordonați pentru aceia dintre angajați care au o vechime
mai mare decât managerii lor.

a. SELECT e.first_name || ' ' || e.last_name AS employee_name, e.hire_date AS employee_hire_date,
m.first_name || ' ' || m.last_name AS manager_name, m.hire_date AS manager_hire_date
FROM employees e
JOIN employees m ON e.manager_id = m.employee_id
WHERE e.hire_date < m.hire_date;

19. Scrieți o interogare care întoarce numele, prenumele și salariul acelor angajați care au
salarii cel mult egale cu cel al lui Popp .

a. SELECT first_name, last_name, salary
FROM employees
WHERE salary <= (SELECT MAX(salary) FROM employees WHERE last_name = 'Popp');

Lab 2

1.1

CREATE TABLE employees_AM_3141a AS SELECT employee_id AS id, first_name AS prenume, last_name AS nume, salary AS salariu FROM employees;

CREATE TABLE employees_AM_3141a AS SELECT employee_id AS id, first_name AS prenume, last_name AS nume, salary AS salariu, HIRE_DATE AS AnAngajare FROM employees;

1.2

```
DESCRIBE employees_AM_3141a;
SELECT * FROM employees;
```

1.3.a

```
INSERT INTO employees_AM_3141a (ID, PRENUME, NUME, SALARIU, AnAngajare)
VALUES (260, 'Mila', 'Jhonson', 12000, TO_DATE('2024-10-08', 'YYYY-MM-DD'));

INSERT INTO employees_AM_3141a (ID, PRENUME, NUME, SALARIU, AnAngajare)
VALUES (261, 'Grace', 'Smith', 8000, TO_DATE('2024-10-08', 'YYYY-MM-DD'));

INSERT INTO employees_AM_3141a (ID, PRENUME, NUME, SALARIU, AnAngajare)
VALUES (262, '', 'Bolt', 4000, TO_DATE('2024-10-08', 'YYYY-MM-DD'));
```

1.4

```
UPDATE employees_AM_3141a SET prenume ='Jonathan' where nume = 'Bolt';
```

1.5

```
SELECT * FROM employees_AM_3141a
WHERE EXTRACT(YEAR FROM ANANGAJARE) = EXTRACT(YEAR FROM
SYSDATE);
```

1.6

```
UPDATE employees_AM_3141a
SET salariu = salariu * 1.10
WHERE salariu = (SELECT MIN(salariu) FROM employees_AM_3141a);

SELECT * FROM employees_AM_3141a;
```

1.7

```
UPDATE employees_AM_3141a
SET salariu = 5500
WHERE salariu < 5500;

SELECT COUNT(*) AS NumarAngajatiBeneficiati
FROM employees_AM_3141a
WHERE salariu = 5500;

SELECT * FROM employees_AM_3141a
WHERE salariu = 5500;
```

1.8.

```
UPDATE employees_AM_3141a
SET salariu = (SELECT AVG(salariu) FROM employees_AM_3141a)
WHERE id = 262;

SELECT * FROM employees_AM_3141a WHERE id = 262;
```

1.9.

```
DELETE FROM employees_AM_3141a
WHERE salariu >= (SELECT AVG(salariu) FROM employees_AM_3141a);
SELECT * FROM employees_AM_3141a
ORDER BY salariu DESC;
```

2.1

```
SELECT department_id, MAX(salary) AS MAXIMUM
FROM employees
GROUP BY department_id;

SELECT &&GROUP_COL, MAX(&NUME_COL) AS MAXIMUM FROM &TABEL GROUP
BY &&GROUP_COL;
```

2.2

```
SELECT first_name, last_name
FROM &TABEL1 a, &TABEL2 b
WHERE &Conditie;
```

2.3

```
SELECT id, prenume || ' ' || nume AS nume_complet, salariu
FROM employees_AM_3141a
WHERE salariu > (SELECT salariu FROM employees_AM_3141a WHERE prenume =
'&PRENUME' AND nume = '&NUME');
```

2.4

```
SELECT UPPER(prenume || ' ' || nume) AS nume_complet, LENGTH(prenume || ' '
|| nume) AS lungime
FROM employees_AM_3141a
WHERE SUBSTR(nume, 1, 2) = UPPER('&LITERE')
ORDER BY nume_complet;
```

```
SELECT UPPER(FIRST_NAME || ' ' || LAST_NAME) AS nume_complet,
LENGTH(FIRST_NAME || ' ' || LAST_NAME) AS lungime
FROM employees
WHERE SUBSTR(LAST_NAME, 1, 2) = UPPER('&Litere')
ORDER BY nume_complet;
```

2.5

```
SELECT &camp1,&camp2,&camp3 FROM &tabel WHERE &cond;
```

camp1:first_name

camp2:last_name

camp3:salary

tabel:employees

cond:salary>5000

3.1

CASE

```
SELECT
first_name,
last_name,
CASE
WHEN job_id = 'AD_PRES' THEN 'A'
WHEN job_id = 'ST_MAN' THEN 'B'
WHEN job_id = 'IT_PROG' THEN 'C'
ELSE '0'
END AS categorie
FROM
employees_AM_3141a;
```

DECODE

```
SELECTfirst_name,last_name, DECODE(job_id, 'AD_PRES', 'A', 'ST_MAN', 'B',
'IT_PROG', 'C', '0')AS categorie FROM employees;
```

3.2

a.

```
SELECT department_id, first_name, last_name, salary,
CASE
```

WHEN department_id = 20 THEN salary * 1.75

WHEN department_id = 40 THEN salary * 2.05

WHEN department_id = 90 THEN salary * 1.02

ELSE salary END AS salariu_nou

FROM employees;

SELECT department_id, first_name, last_name, salary, DECODE(department_id, 20, salary * 1.75, 40, salary * 2.05, 90, salary * 1.02, salary) AS salariu_nou FROM employees;

PBD Lab-3

1.

Comparare împerecheată

```
SELECT e.first_name || ' ' || e.last_name AS employee_name,
e.department_id, e.salary
FROM employees e
WHERE (e.department_id, e.salary) IN (
  SELECT department_id, salary
  FROM employees
  WHERE commission_pct IS NOT NULL
);
```

Comparare neîmperecheată

```
SELECT e.first_name || ' ' || e.last_name AS employee_name,
e.department_id, e.salary
FROM employees e
WHERE e.department_id IN (
  SELECT department_id
  FROM employees
  WHERE commission_pct IS NOT NULL
```

```
)
AND e.salary IN (
  SELECT salary
  FROM employees
  WHERE commission_pct IS NOT NULL
);
```

2.

```
SELECT e.first_name || ' ' || e.last_name AS employee_name,
e.hire_date, e.salary, e.commission_pct
FROM employees e
WHERE e.salary IN (
  SELECT salary
  FROM employees
  WHERE last_name = 'King'
)
AND NVL(e.commission_pct,0) IN (
  SELECT NVL(commission_pct,0)
  FROM employees
  WHERE last_name = 'King'
);
```

3.

```
SELECT e.first_name || ' ' || e.last_name AS employee_name,
       d.department_name,
       e.salary
FROM employees e
JOIN departments d ON e.department_id = d.department_id
WHERE (e.salary,NVL(e.commission_pct,0)) IN (
  SELECT salary,NVL(commission_pct,0)
  FROM employees
  WHERE department_id IN (
```

```
    SELECT department_id
    FROM departments
    WHERE location_id = 1700
  )
);
```

4

```
SELECT d.department_name, emp.num_employees, d.location_id
FROM departments d, (
  SELECT department_id, COUNT(*) AS num_employees
  FROM employees
  GROUP BY department_id
) emp
WHERE d.department_id = emp.department_id;
```

5.

```
SELECT e.first_name, e.last_name,
  CASE
    WHEN e.salary > (
      SELECT MAX(salary)
      FROM employees
      WHERE department_id = 50
    ) THEN 'mai mare'
    ELSE 'mai mic'
  END AS salary_comparison
FROM employees e;
```

6.

```
SELECT first_name || ' ' || last_name AS employee_name
FROM employees e
```

```
WHERE salary < (
    SELECT AVG(salary)
    FROM employees
    WHERE department_id = e.department_id
);
```

7.

```
SELECT e.first_name, e.last_name
FROM employees e
WHERE EXISTS (
    SELECT 1
    FROM employees c
    WHERE c.department_id = e.department_id
    AND c.hire_date > e.hire_date
    AND c.salary > e.salary
);
```

8.

```
SELECT e.first_name, e.last_name
FROM employees e
WHERE EXISTS (
    SELECT 1
    FROM departments d
    WHERE d.manager_id = e.employee_id
);
SELECT e.first_name, e.last_name
FROM employees e
JOIN departments d ON e.employee_id = d.manager_id;
```

9.

```
SELECT e.first_name, e.last_name
FROM employees e
WHERE NOT EXISTS (
    SELECT 1
    FROM job_history jh
    WHERE jh.employee_id = e.employee_id
);
```

10.

```
WITH max_salary_company AS (
    SELECT MAX(salary) AS max_salary FROM employees
),
job_max_salary AS (
    SELECT job_id, MAX(salary) AS max_job_salary
    FROM employees
    GROUP BY job_id
)
SELECT j.job_title
FROM jobs j
JOIN job_max_salary jms ON j.job_id = jms.job_id
JOIN max_salary_company msc ON jms.max_job_salary > msc.max_s
alary / 2;
```

11.

```
WITH job_max_salary AS (
    SELECT job_id, MAX(salary) AS max_job_salary
    FROM employees
    GROUP BY job_id
)
SELECT e.first_name || ' ' || e.last_name AS nume_complet,
       e.salary,
```

```
       (jms.max_job_salary - e.salary) AS diferenta_salariu
FROM employees e
JOIN job_max_salary jms ON e.job_id = jms.job_id;
```

PDB Lab-4

1.

```
CREATE TABLE employees_AM AS
SELECT * FROM employees;
ALTER TABLE employees_AM
ADD functie VARCHAR2(255);
UPDATE employees_AM e
SET e.functie = (SELECT j.job_title
                 FROM jobs j
                 WHERE j.job_id = e.job_id);
```

1.

```
ALTER TABLE employees_AM
ADD departament VARCHAR2(35);
UPDATE employees_AM e
SET functie = (SELECT j.job_title
               FROM jobs j
               WHERE e.job_id = j.job_id),
    departament = (SELECT d.department_name
                   FROM departments d
                   WHERE e.department_id = d.department_id);
```

1.

```
CREATE TABLE job_history_AM AS
SELECT * FROM job_history;
DELETE FROM job_history_AM jh
```

```
WHERE NOT EXISTS (SELECT 1
                  FROM employees_AM e
                  WHERE jh.employee_id = e.employee_id);
```

1.

```
DELETE FROM employees_AM e
WHERE e.department_id IN (SELECT d.department_id
                          FROM departments d
                          JOIN locations l ON d.location_id =
l.location_id
                          WHERE l.city = 'Seattle');
```

1.

```
CREATE TABLE nivel_salarizare_AM(
nivel_sal VARCHAR2(20),
    min_sal NUMBER(8,2),
    max_sal NUMBER(8,2)
);

INSERT INTO nivel_salarizare_AM (nivel_sal, min_sal, max_sal)
VALUES ('Junior', 2000, 4000);
INSERT INTO nivel_salarizare_AM (nivel_sal, min_sal, max_sal)
VALUES ('Middle', 4001, 6000);
INSERT INTO nivel_salarizare_AM (nivel_sal, min_sal, max_sal)
VALUES ('Senior', 6001, 8000);
INSERT INTO nivel_salarizare_AM (nivel_sal, min_sal, max_sal)
VALUES ('Expert', 8001, 10000);

SELECT e.first_name, e.last_name, e.salary, n.nivel_sal
FROM employees e
JOIN nivel_salarizare_AM n
ON e.salary BETWEEN n.min_sal AND n.max_sal;
```

1.

```
SELECT n.nivel_sal, COUNT(e.employee_id) AS numar_angajati
FROM employees e
JOIN nivel_salarizare_AM n
ON e.salary BETWEEN n.min_sal AND n.max_sal
GROUP BY n.nivel_sal;
```

1.

```
SELECT e.first_name ||' '|| e.last_name AS
nume_comp, e.salary, n.nivel_sal, j.job_title
FROM employees e
JOIN nivel_salarizare_AM n
ON e.salary
BETWEEN n.min_sal AND n.max_sal
JOIN jobs j ON e.job_id = j.job_id
WHERE e.department_id = 50;
```

PBD Lab-5

1. Rulați următorul script și specificați ce reprezintă el.

   ```
   SELECT employee_id, first_name || ' ' || last_name as anga
   jat, job_id
   FROM employees
   UNION
   SELECT employee_id, 'NULL' as angajat, job_id
   FROM job_history;
   ```

   Puteți obține același rezultat folosind, spre exemplu, un outer join? Încercați și comentați
   rezultatul.

```
SELECT employee_id, first_name || ' ' || last_name as anga
jat, job_id
FROM employees
UNION
SELECT employee_id, 'NULL' as angajat, job_id
FROM job_history;

SELECT e.employee_id,
       COALESCE(e.first_name || ' ' || e.last_name, 'NUL
L') AS angajat,
       COALESCE(e.job_id, jh.job_id) AS job_id
FROM employees e
FULL OUTER JOIN job_history jh
ON e.employee_id = jh.employee_id;
```

FULL OUTER JOIN returnează toate rândurile din ambele tabele, completând cu NULL valorile care lipsesc. Funcția COALESCE se folosește pentru a înlocui valorile NULL cu ‘NULL’ sau alte valori din celălalt set de date. Rezultatul va fi similar celui obținut prin UNION.

2. Utilizând operatorii pe mulțimi, scrieți o interogare care să afișeze identificatorii acelor
departamente care nu conțin job-ul cu identificatorul "ST_CLERK".

```
SELECT department_id
FROM departments d
WHERE department_id NOT IN (
    SELECT department_id
    FROM employees
    WHERE job_id = 'ST_CLERK'
);
```

3. Scrieți o interogare care sa afișeze identificatorii departamentelor care au cel puțin un
angajat.

```
SELECT department_id
FROM employees
GROUP BY department_id;
```

4. Folosind operatorii pe mulțimi găsiți joburile din departamentul 50 care nu sunt și în
departamentul 80.

```
SELECT job_id
FROM employees
WHERE department_id = 50
MINUS
SELECT job_id
FROM employees
WHERE department_id = 80;
```

5. Scrieți o listă a job-urilor pentru departamentele 10, 50, 40 în această ordine. Afișați
identificatorii job-urilor și identificatorii departamentelor folosindu-vă de operatorii pe
mulțimi.

```
-- Job-uri pentru departamentul 10
SELECT job_id, department_id
FROM employees
WHERE department_id = 10

UNION ALL

-- Job-uri pentru departamentul 50
SELECT job_id, department_id
FROM employees
WHERE department_id = 50

UNION ALL
```

```
-- Job-uri pentru departamentul 40
SELECT job_id, department_id
FROM employees
WHERE department_id = 40;
```

6. Se cere un (singur) raport cu următoarele date:

   a. Numele angajatului și identificatorul departamentelor pentru toți angajații, indiferent dacă aceștia aparțin sau nu unui departament ;

```
SELECT e.first_name || ' ' || e.last_name AS angajat, e.de
partment_id
FROM employees e
LEFT JOIN departments d ON e.department_id = d.department_
id;
```

   b. Identificatorul și numele departamentului chiar dacă acesta nu are încă angajați.

```
SELECT d.department_id, d.department_name
FROM departments d
LEFT JOIN employees e ON d.department_id = e.department_i
d;
```

7. Studiați exemplul următor și spuneți ce afișează :

```
SELECT job_id, salary, null "totsal salarii pe job"
FROM employees
UNION
SELECT job_id, null, sum(salary)
FROM employees
GROUP BY job_id
UNION
```

```
SELECT null, null, sum(salary)
FROM employees;
```

După modelul studiat scrieți O INTEROGARE care să afișeze :
• numărul de angajați din fiecare departament și pe fiecare job;
• numărul de angajați din fiecare departament;
• numărul de angajați pe fiecare job
• numărul total de angajați.

Interogarea care îndeplinește cerințele

```
-- Numărul de angajați din fiecare departament și pe fiecare
job
SELECT department_id, job_id, COUNT(*) AS num_angajati, null
AS "totsal salarii pe job"
FROM employees
GROUP BY department_id, job_id

UNION

-- Numărul de angajați din fiecare departament
SELECT department_id, null AS job_id, COUNT(*) AS num_angajat
i, null AS "totsal salarii pe job"
FROM employees
GROUP BY department_id

UNION

-- Numărul de angajați pe fiecare job
SELECT null AS department_id, job_id, COUNT(*) AS num_angajat
i, null AS "totsal salarii pe job"
FROM employees
GROUP BY job_id

UNION
```

```
-- Numărul total de angajați
SELECT null AS department_id, null AS job_id, COUNT(*) AS num
_angajati, null AS "totsal salarii pe job"
FROM employees;
```

1. Folosind unul din operatorii pe mulțimi creați un raport precum cel de mai jos :
   Rezultat
   1
   3
   2
   5
   7

   ```
   SELECT 1 FROM dual
   UNION
   SELECT 3 FROM dual
   UNION
   SELECT 2 FROM dual
   UNION
   SELECT 5 FROM dual
   UNION
   SELECT 7 FROM dual;
   ```

2. Faceți o copie a tabelului employees, numiți-l employees_id (id inițialele voastre) și pentru
   angajații din Toronto (location_id=1800) și Southlake (location_id=1400):

a. Se setează identificatorul departmentului pentru acești angajați ca fiind egal cu identificatorul departmentului corespunzător orașului Londra (location_id=2400).

```
UPDATE employees_AM e
SET department_id = (
    SELECT d.department_id
    FROM departments d
    WHERE d.location_id = 2400
)
```

```
WHERE e.department_id IN (
    SELECT d.department_id
    FROM departments d
    WHERE d.location_id IN (1800, 1400)
);
```

b. Se setează pentru acesti angajați salariul ca fiind de 1,2 * salariul mediu corespunzător departamentului din care face parte fiecare.

```
UPDATE employees_AM e
SET salary = 1.2 * (
    SELECT AVG(e2.salary)
    FROM employees_AM e2
    WHERE e2.department_id = e.department_id
)
WHERE e.department_id IN (
    SELECT d.department_id
    FROM departments d
    WHERE d.location_id IN (1800, 1400)
);
```

c. Se setează comisionul fiecărui angajat ca fiind egal cu 1,5 * comisionul mediu corespunzător departamentului din care face parte fiecare.

```
UPDATE employees_AM e
SET commission_pct = 1.5 * (
    SELECT AVG(e2.commission_pct)
    FROM employees_AM e2
    WHERE e2.department_id = e.department_id
)
WHERE e.department_id IN (
    SELECT d.department_id
    FROM departments d
    WHERE d.location_id IN (1800, 1400)
);
```

Câte înregistrări au fost modificate?

7 rows updated.
0 rows updated.
0 rows updated.

1. Afișati care sunt angajații care au suferit modificări la exercițiul anterior folosindu-vă de
   operatorii pe mulțimi?

```
SELECT employee_id, first_name, last_name
FROM employees_AM
MINUS
SELECT employee_id, first_name, last_name
FROM employees;
```

1. Afișati aceiași angajați care au suferit modificări folosindu-vă de join.

```
SELECT e1.employee_id, e1.first_name, e1.last_name
FROM employees e1
JOIN employees_AM e2 ON e1.employee_id = e2.employee_id
WHERE e1.salary <> e2.salary
   OR e1.department_id <> e2.department_id
   OR e1.commission_pct <> e2.commission_pct;
```

1. Realizați un script folosind operatorii pe mulțimi pentru a afișa locațiile comune din
   locations și departments. Cum puteți afișa și numele locațiilor (city)?

```
SELECT l.location_id, l.city
FROM locations l
JOIN departments d ON l.location_id = d.location_id;
```

1. Crearea tabelelor AM_tableno1 și AM_tableno2

```
CREATE TABLE AM_tableno1 (id NUMBER);
CREATE TABLE AM_tableno2 (id NUMBER);
INSERT INTO AM_tableno1 (id) VALUES (1);
INSERT INTO AM_tableno1 (id) VALUES (2);
INSERT INTO AM_tableno1 (id) VALUES (3);
INSERT INTO AM_tableno1 (id) VALUES (4);
INSERT INTO AM_tableno1 (id) VALUES (5);
INSERT INTO AM_tableno1 (id) VALUES (6);
INSERT INTO AM_tableno1 (id) VALUES (7);
INSERT INTO AM_tableno1 (id) VALUES (8);
INSERT INTO AM_tableno1 (id) VALUES (9);
INSERT INTO AM_tableno1 (id) VALUES (10);
INSERT INTO AM_tableno2 (id) VALUES (4);
INSERT INTO AM_tableno2 (id) VALUES (5);
INSERT INTO AM_tableno2 (id) VALUES (6);
INSERT INTO AM_tableno2 (id) VALUES (7);
INSERT INTO AM_tableno2 (id) VALUES (8);
INSERT INTO AM_tableno2 (id) VALUES (9);
INSERT INTO AM_tableno2 (id) VALUES (10);
INSERT INTO AM_tableno2 (id) VALUES (11);
INSERT INTO AM_tableno2 (id) VALUES (12);
```

a. Aplicați operatorii pe mulțimi pe cele 2 tabele.
UNION, UNION ALL, MINUS, INTERSECT

```
SELECT id FROM AM_tableno1
UNION
SELECT id FROM AM_tableno2;

SELECT id FROM AM_tableno1
UNION ALL
SELECT id FROM AM_tableno2;

SELECT id FROM AM_tableno1
MINUS
```

```
SELECT id FROM AM_tableno2;

SELECT id FROM AM_tableno1
INTERSECT
SELECT id FROM AM_tableno2;
```

b. Aplicați JOIN-urile pe cele 2 tabele.
(INNER, OUTER, LEFT, RIGHT)
SELECT
a.id, b.id
FROM AM_tableno1 a
INNER JOIN AM_tableno2 b ON
a.id = b.id;

SELECT a.id, b.id
FROM AM_tableno1 a
LEFT JOIN AM_tableno2 b ON
a.id = b.id;

SELECT a.id, b.id
FROM AM_tableno1 a
RIGHT JOIN AM_tableno2 b ON
a.id = b.id;

SELECT a.id, b.id
FROM AM_tableno1 a
FULL OUTER JOIN AM_tableno2 b ON
a.id = b.id;

PBD Lab-6

Întrebări

a. Care sunt privilegiile uzuale ale unui DBA? (Ce înseamnă DBA?)

DBA este acronimul pentru Database Administrator (Administrator al Bazei de Date). Privilegiile tipice ale unui DBA includ:

CREATE/ALTER/DROP USER: gestionarea utilizatorilor și a drepturilor acestora.

GRANT/REVOKE: atribuirea și retragerea privilegiilor.

BACKUP/RECOVERY: realizarea copiilor de siguranță și restaurarea bazei de date.

CREATE/ALTER/DROP TABLE, VIEW, INDEX: gestionarea obiectelor bazei de date.

SELECT ANY TABLE, INSERT ANY TABLE: acces la date și modificarea acestora în toate tabelele.

RESOURCE MANAGEMENT: gestionarea resurselor pentru utilizatori și aplicații.

b. Ce privilegii poate să posede un user normal?

Un utilizator normal poate avea privilegii de acces la obiecte (de exemplu, SELECT, INSERT, UPDATE, DELETE pe anumite tabele). De asemenea, poate avea drepturi limitate de creare a obiectelor în propria schemă (de exemplu, CREATE TABLE, CREATE VIEW).

c. Care sunt posibilele privilegii asupra obiectelor schemă?

Posibilele privilegii asupra obiectelor includ:

SELECT: acces la datele dintr-un obiect (tabel, vedere).

INSERT, UPDATE, DELETE: modificarea datelor din obiect.

ALTER: modificarea structurii obiectului (de exemplu, adăugarea coloanelor).

INDEX: crearea de indici pe tabel.

EXECUTE: pentru apelarea funcțiilor sau procedurilor stocate.

REFERENCES: crearea de constrângeri FOREIGN KEY care fac referință la tabelul respectiv.

d. În ce scop sunt create profilele?

Profilele sunt create pentru a defini restricții și limite legate de resursele utilizate de un cont, pentru a gestiona autentificarea și securitatea, cum ar fi:

Numărul maxim de tentative de conectare nereușite.

Timpul maxim de inactivitate.

Blocarea automată a conturilor după un anumit număr de încercări eșuate de autentificare.

Limite de utilizare a resurselor (CPU, sesiuni etc.).

e. Ce este un domeniu de securitate?

Domeniul de securitate (sau security domain) reprezintă totalitatea privilegiilor, profilurilor și resurselor la care un utilizator are acces. Definirea acestuia asigură că utilizatorul are acces doar la acele resurse necesare.

f. Ce privilegiu trebuie să aibă un utilizator pentru a se putea conecta la o baza de date? Ce tip de privilegiu este acesta?

Utilizatorul trebuie să aibă privilegiul CREATE SESSION. Este un privilegiu de sistem, necesar pentru a iniția o sesiune de conectare.

g. Se presupune că sunteți un DBA și creați mai mulți useri care solicită aceleași privilegii. Ce ar trebui să faceți pentru a vă ușura munca?

Ar trebui să creați un rol care să includă toate privilegiile necesare și să atribuiți acest rol fiecărui utilizator. Rolurile simplifică gestionarea privilegiilor comune pentru mai mulți utilizatori.

h. În cazul unui obiect al unei baze de date, cine poate transmite privilegii în afară de proprietarul său?

În afară de proprietarul obiectului, utilizatorii cărora le-a fost acordat privilegiul cu opțiunea WITH GRANT OPTION pot transmite aceste privilegii altor utilizatori.

i. Considerați că unui rol îi poate fi atribuit un alt rol? Explicați răspunsul.

Da, unui rol i se poate atribui un alt rol. Aceasta este o funcționalitate utilă în ierarhizarea drepturilor, permițând combinarea privilegiilor din mai multe roluri într-unul singur pentru a gestiona mai eficient accesul și securitatea.

Exerciții

1. Creați un cont utilizator nou, care să se identifice prin parolă. Atât numele userului cât
   și parola le stabiliți voi.

```
CREATE USER aydgn INDENTIFIED BY Student;
```

1. Deconectați-vă de la baza de date și încercați să vă reconectați folosind contul nou
   creat. Ce se întâmplă?

```
user AYDGN lacks CREATE SESSION privilege; logon denied
```

2. Reconectați-vă la baza de date cu userul hr și atribuiți privilegiul necesar pentru a vă
   putea conecta cu noul utilizator la baza de date. Reconectați-vă cu noul user și verificați
   privilegiile pe care le are acest utilizator.

```
GRANT CREATE SESSION TO aydgn;
SELECT * FROM SESSION_PRIVS;
```

3. Reconectați-vă la baza de date cu userul hr. Atribuiți user-ului nou creat rolurile
   CONNECT, RESOURCE, SELECT ANY DICTIONARY, SELECT ANY TABLE și CREATE TABLE
   după care încercați din nou să vă conectați, cu acesta la baza de date. Verificați
   privilegiile pe care le are acum userul nou creat.

```
GRANT CONNECT, RESOURCE, SELECT ANY DICTIONARY, SELECT ANY
TABLE, CREATE TABLE TO aydgn;
```

4. Verificați dacă s-a creat schema userului nou folosind instrucțiunea:

```
SELECT DISTINCT OWNER FROM DBA_TABLES;
Sau
SELECT * FROM DBA_TABLES where UPPER(OWNER) = UPPER('aydg
n');
```

5. Conectați-vă cu userul nou creat. Încercați să interogați tabelul employees. Ce observați?

```
SELECT * FROM hr.employees;
- Ar trebui sa functioneze datorita SELECT ANY TABLE.
```

6. Creați un tabel identic cu tabelul EMPLOYEES din schema userului HR. Pentru a adresa
   obiecte din schema altui user se folosește notația dot(.) (ex.pentru a adresa tabelul jobs
   din schema userului HR se folosește notația hr.jobs)

```
CREATE TABLE employees AS SELECT * FROM hr.employees;
```

7. Reconectați-vă la baza de date cu userul hr și revocați privilegul SELECT ANY TABLE
   userului nou creat.

```
REVOKE SELECT ANY TABLE FROM aydgn;
```

8. Conectați-vă cu userul nou creat și încercați apoi să creați un tabel identic cu JOB_HISTORY din schema HR. Ce observați?

```
CREATE TABLE job_history AS SELECT * FROM hr.job_history;
```

9. Verificați din nou dacă s-a creat schema userului nou folosind instrucțiunile de la
   punctul 5. Ce observați?

```
SELECT DISTINCT OWNER FROM DBA_TABLES;
```

10. Creați un rol cu numele vostru și asignați-i privilegii care să îi permita crearea de tabele
    și de vederi în orice schemă.

```
CREATE ROLE aydgn_role;
GRANT CREATE ANY TABLE, CREATE ANY VIEW TO aydgn_role;
GRANT aydgn_role TO aydgn;
```

11. Atribuiți acest rol userului creat la punctul 2, și verificați dacă puteți crea în schema user-

ului SCOTT un tabel care să fie, de asemenea copie a tabelului EMPLOYEES din schema HR.

```
CREATE TABLE scott.employees AS SELECT * FROM hr.employees
ERROR at line 1:
ORA-00942: table or view does not exist
```

12. Creați un profil care să limiteze la 3 numărul încercărilor de conectare pentru care se
admite tastarea greșită a parolei. După aceste încercări contul va fi blocat pentru 5 zile.

```
CREATE PROFILE profil_provizoriuAM LIMIT FAILED_LOGIN_ATTE
MPTS 3 PASSWORD_LOCK_TIME 5;
ALTER USER aydgn PROFILE profil_provizoriuAM;
```

Atribuiți userului nou profilul creat. Verificați dacă user-ul se blochează după 3 tastări
greșite ale parolei.

13. Pentru fiecare din problemele anterioare specificați userul cu care ați fost conectați, într-
un tabel de forma:

Nr. exercițiu	Utilizator conectat
1	SYS/ SYSTEM
2	aydgn
3	HR
4	HR
5	HR
6	aydgn
7	aydgn
8	HR
9	aydgn

10	HR
11	SYS/ SYSTEM
12	aydgn
13	HR
14	NULL
15	SYS/ SYSTEM

14. Ștergeți user-ul creat.