Лабораторна робота 10  
«Ієрархічні запити»  
⚠️ Для завдань/розділів відмічених «📷» – скріншот – обов’язково. Для інших – за домовленістю із викладачем.  
Завдання лабораторної роботи виконую всі

Завдання роботи  
1. 📷 Напишіть запит, який повертав би з таблиці emp інформацію:

* про ім'я співробітника;
* про рівень підпорядкованості (найвищий рівень – головний начальник, який нікому не підпорядковується – рівень 1);
* про шлях підзвітності у форматі: /керівник 1/керівник 2/рядовий співробітник.

Звіт має бути відсортований за рівнем підпорядкованості.

WITH RECURSIVE cte\_query AS (

SELECT e.\*, e.ename::varchar as "path", 1 as "level"

FROM emp e

WHERE e.mgr is null

UNION

SELECT e.\*, "path" || '/' || e.ename, level+1

FROM cte\_query

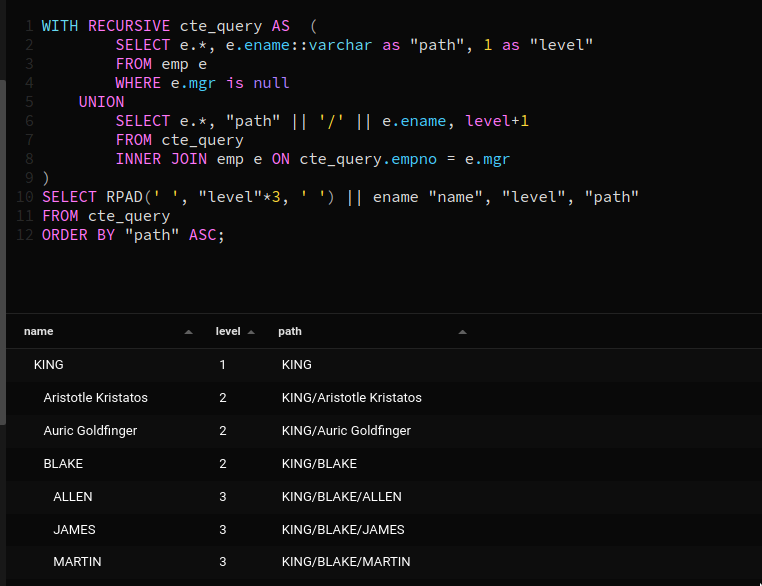
INNER JOIN emp e ON cte\_query.empno = e.mgr

)

SELECT RPAD(' ', "level"\*3, ' ') || ename "name", "level", "path"

FROM cte\_query

ORDER BY "path" ASC;



2. Виведіть усі імена та зарплати начальників співробітника на ім'я «JET LI»

WITH RECURSIVE cte\_query AS (

SELECT e.\* FROM emp e

WHERE empno = (SELECT e1.mgr FROM emp e1 WHERE e1.ename = 'JET LI')

UNION

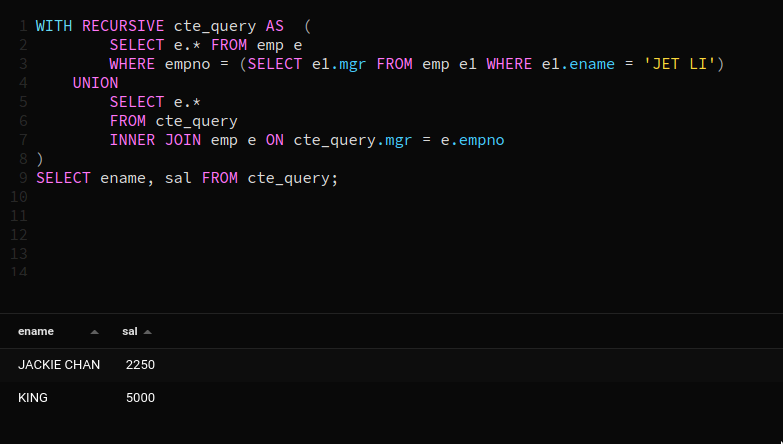
SELECT e.\*

FROM cte\_query

INNER JOIN emp e ON cte\_query.mgr = e.empno

)

SELECT ename, sal FROM cte\_query;



3. Хто з начальників співробітника «JET LI» отримує найвищу зарплату?

WITH RECURSIVE cte\_query AS (

SELECT e.\* FROM emp e

WHERE empno = (SELECT e1.mgr FROM emp e1 WHERE e1.ename = 'JET LI')

UNION

SELECT e.\*

FROM cte\_query

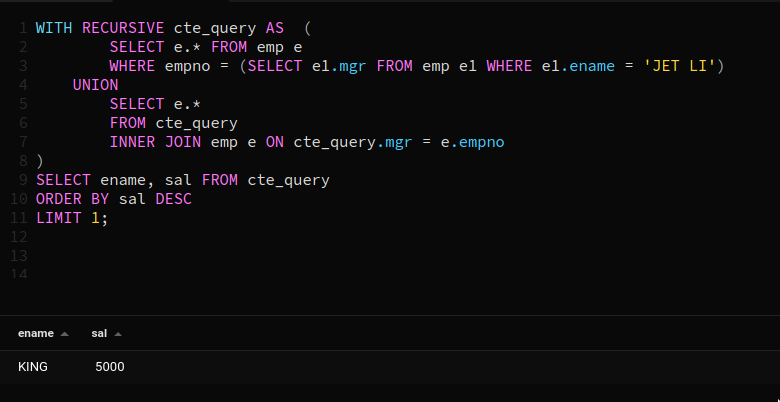
INNER JOIN emp e ON cte\_query.mgr = e.empno

)

SELECT ename, sal FROM cte\_query

ORDER BY sal DESC

LIMIT 1;



4. \*Виведіть клерків, які є підлеглими (чи підлеглими підлеглих, ...) Блейка.

WITH RECURSIVE cte\_query AS (

SELECT e.\* FROM emp e WHERE ename = 'BLAKE'

UNION

SELECT e.\*

FROM cte\_query

INNER JOIN emp e ON cte\_query.empno = e.mgr AND e.job = 'CLERK'

)

SELECT ename, sal FROM cte\_query

WHERE ename != 'BLAKE';



5. \*\* Виведіть клерків, які НЕ є підлеглими (чи підлеглими підлеглих, ...) Блейка.

WITH RECURSIVE cte\_query AS (

SELECT e.\* FROM emp e WHERE mgr is null

UNION

SELECT e.\*

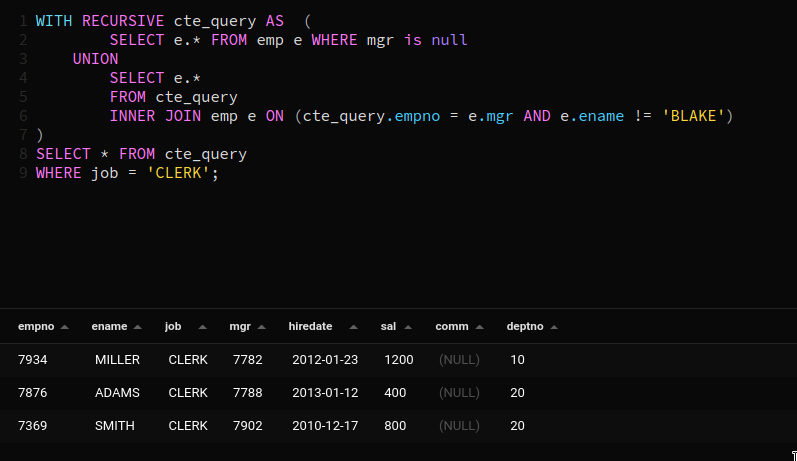
FROM cte\_query

INNER JOIN emp e ON (cte\_query.empno = e.mgr AND e.ename != 'BLAKE')

)

SELECT \* FROM cte\_query

WHERE job = 'CLERK';



6. \*\* У кого зі співробітників 2-ї ланки (підлеглих Кінга) у підпорядкуванні найбільше клерків?

WITH RECURSIVE cte\_query AS (

SELECT e.\*, NULL::varchar as "manager", 1 as "level"

FROM emp e

WHERE e.mgr is null

UNION

SELECT e.\*, (

CASE WHEN level = 2 THEN cte\_query.ename

ELSE cte\_query.manager

END

) as "manager", level+1

FROM cte\_query

INNER JOIN emp e ON cte\_query.empno = e.mgr

)

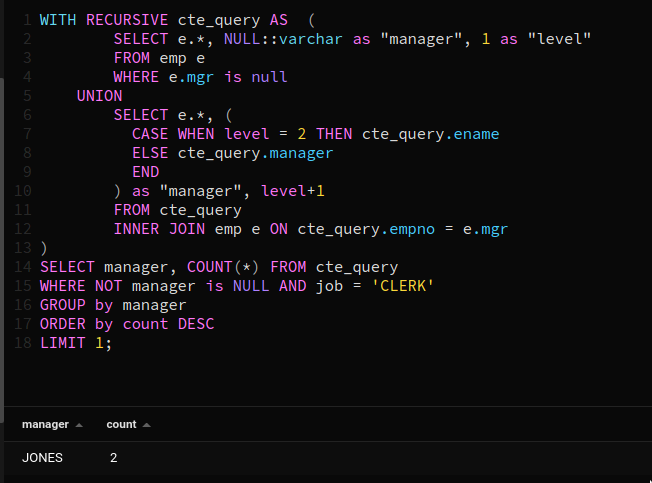
SELECT manager, COUNT(\*) FROM cte\_query

WHERE NOT manager is NULL AND job = 'CLERK'

GROUP by manager

ORDER by count DESC

LIMIT 1;



Пропуски  
7. Завдання

* Створіть таблицю T заповнену числами -3..10

CREATE TABLE "T" AS

WITH RECURSIVE t\_table AS (

SELECT -3 as "num"

UNION

SELECT num+1 FROM t\_table WHERE num < 10

)

SELECT \* FROM t\_table;



* Видаліть з таблиці числа 4 і 6. Уявімо, що ми не знаємо які саме числа були видалені, але точно НЕ перше чи останнє.



* Напишіть запит, який виведе видалені числа.

WITH RECURSIVE t\_table AS (

SELECT -3 as "num"

UNION

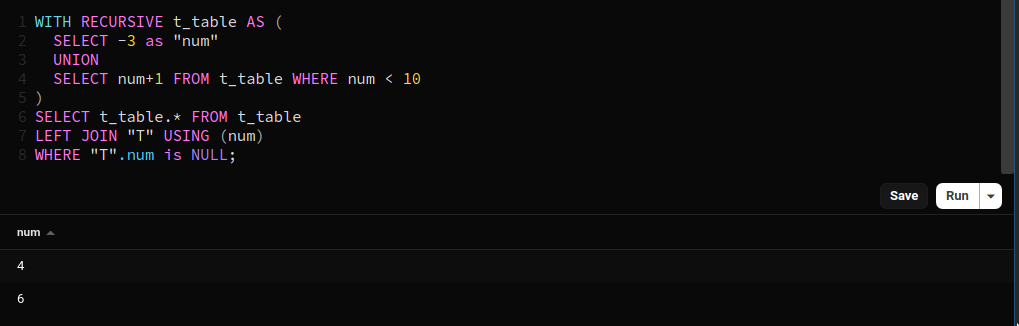
SELECT num+1 FROM t\_table WHERE num < 10

)

SELECT t\_table.\* FROM t\_table

LEFT JOIN "T" USING (num)

WHERE "T".num is NULL;



* \* Як буде виглядати запит, якщо максимальний елемент наперед не відомий

WITH RECURSIVE t\_table AS (

SELECT -3 as "num"

UNION

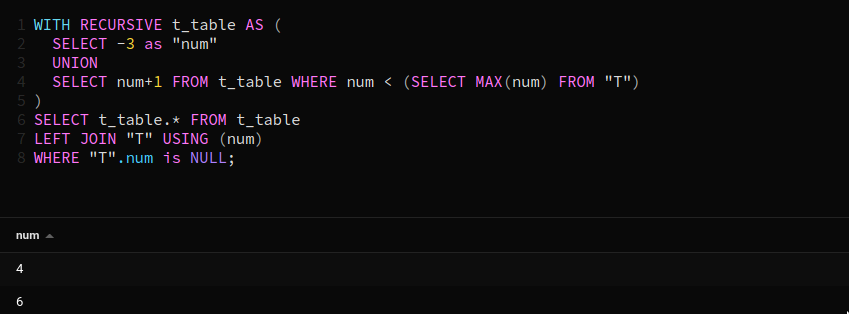
SELECT num+1 FROM t\_table WHERE num < (SELECT MAX(num) FROM "T")

)

SELECT t\_table.\* FROM t\_table

LEFT JOIN "T" USING (num)

WHERE "T".num is NULL;



* \* Як буде виглядати запит, якщо максимальний і мінімальний елемент наперед невідомі?

WITH RECURSIVE t\_table AS (

SELECT (SELECT MIN(num) FROM "T") as "num"

UNION

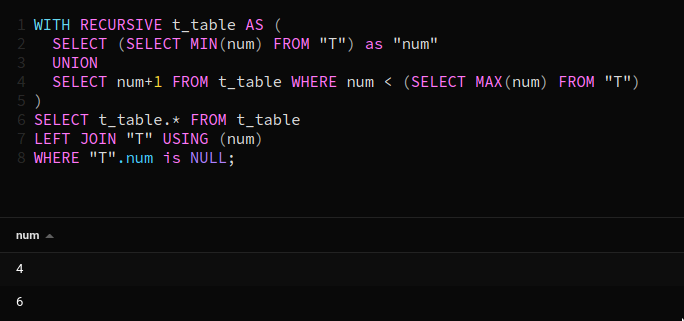
SELECT num+1 FROM t\_table WHERE num < (SELECT MAX(num) FROM "T")

)

SELECT t\_table.\* FROM t\_table

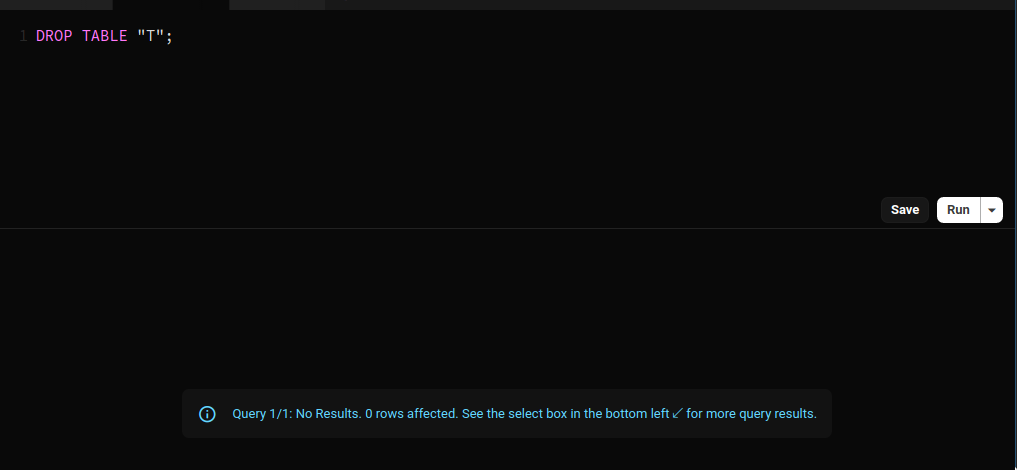
LEFT JOIN "T" USING (num)

WHERE "T".num is NULL;



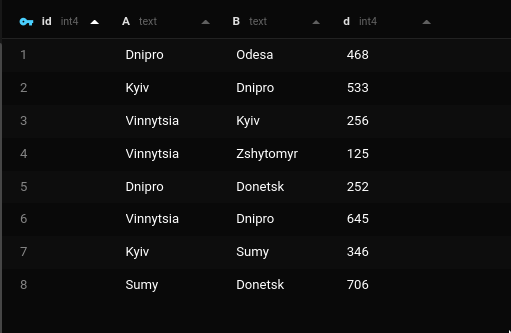
* Видаліть таблицю Т.

DROP TABLE "T";



Втекти звідси  
Створіть таблицю відстаней між містами (не менше 8 рядків),

8. 📷 Виведіть данні з Вашої таблиці.



9. Куди можна доїхати за 2 пересадки з обраного вами міста?

WITH RECURSIVE t\_path AS (

SELECT

"A" as "path",

0 as "level",

"A" as "prev",

0 as "dist"

FROM path WHERE "A" = 'Vinnytsia'

UNION

SELECT

t\_path."path" || ' -> ' || path."B",

"level"+1,

path."B" as "prev",

dist + path.d

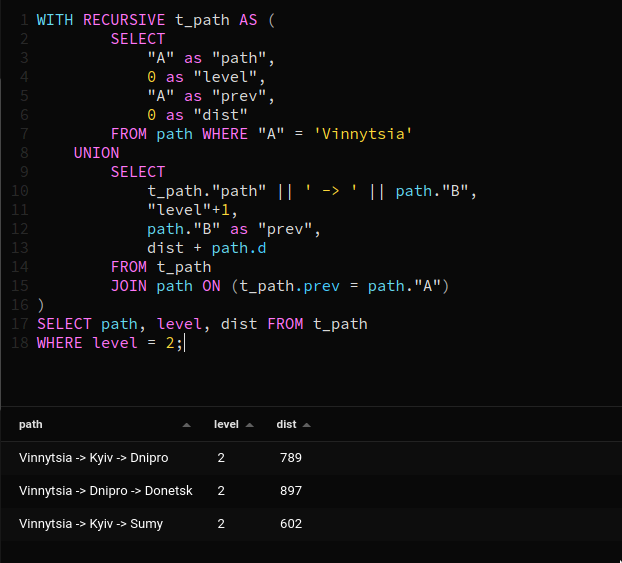
FROM t\_path

JOIN path ON (t\_path.prev = path."A")

)

SELECT path, level, dist FROM t\_path

WHERE level = 2;



10. \* Знайдіть маршрут з мінімальною кількістю пересадок між двома вибраними вами містами.

WITH RECURSIVE t\_path AS (

SELECT

"A" as "path",

0 as "level",

"A" as "last",

0 as "dist"

FROM path WHERE "A" = 'Vinnytsia'

UNION

SELECT

t\_path."path" || ' -> ' || path."B",

"level"+1,

path."B" as "last",

dist + path.d

FROM t\_path

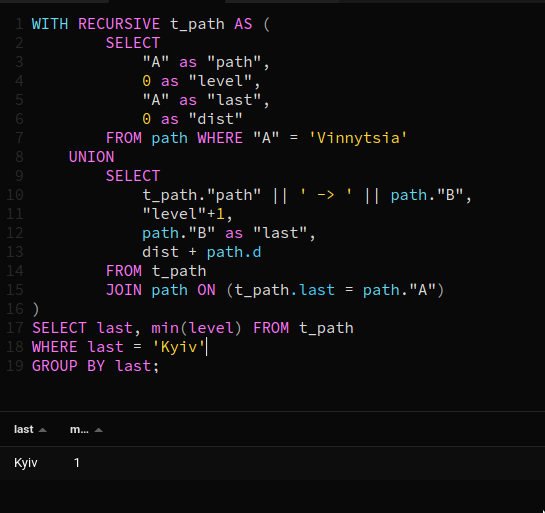
JOIN path ON (t\_path.last = path."A")

)

SELECT last, min(level) FROM t\_path

WHERE last = 'Kyiv'

GROUP BY last;



11. \*\* Знайдіть найкоротший маршрут між вибраними вами містами.

WITH RECURSIVE t\_path AS (

SELECT

"A" as "path",

0 as "level",

"A" as "last",

0 as "dist"

FROM path WHERE "A" = 'Vinnytsia'

UNION

SELECT

t\_path."path" || ' -> ' || path."B",

"level"+1,

path."B" as "last",

dist + path.d

FROM t\_path

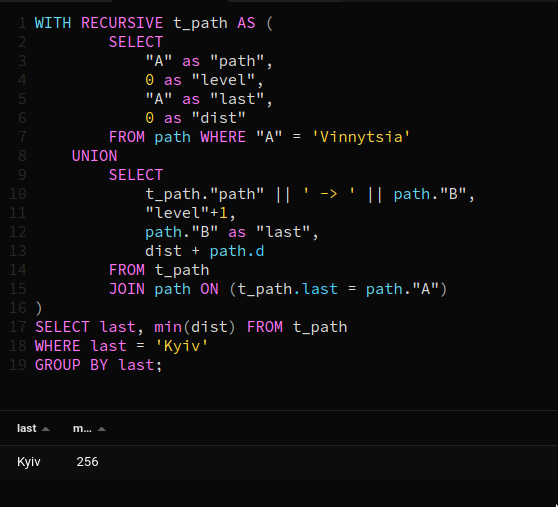
JOIN path ON (t\_path.last = path."A")

)

SELECT last, min(dist) FROM t\_path

WHERE last = 'Kyiv'

GROUP BY last;



Завдання за варіантами  
По завданню 2. Для «А» та «В» їх «спільним найближчим керівником» є «Г»

Варіант 1 📷  
1. \* У співробітниках додайте поле «керівник» - хто з співробітників є керівником для даного. У випадку, якщо клієнт незадоволений швидкістю проведення операції він може звернутись до керівника. Значи операцію виведіть весь перелік керівників – до кого можна звернутись із скаргою.

WITH t\_transactions AS (

SELECT \* FROM transactions WHERE transactionid = 6

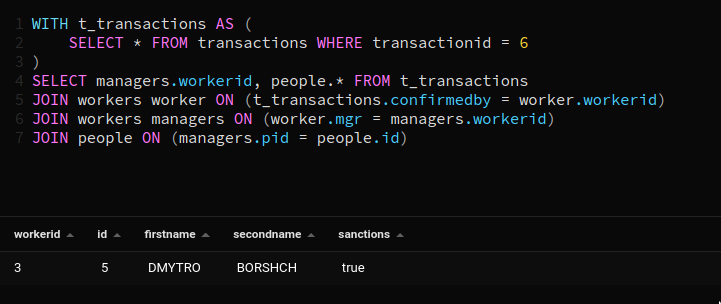
)

SELECT managers.workerid, people.\* FROM t\_transactions

JOIN workers worker ON (t\_transactions.confirmedby = worker.workerid)

JOIN workers managers ON (worker.mgr = managers.workerid)

JOIN people ON (managers.pid = people.id)



2. \*\* Для двох співробітників визначить, хто є їх спільним найближчим керівником

WITH RECURSIVE first\_worker AS (

SELECT \*, 0 as level FROM workers WHERE workers.workerid = 4

UNION

SELECT workers.\*, level + 1 FROM workers

JOIN first\_worker ON (workers.workerid = first\_worker.mgr)

),

second\_worker AS (

SELECT \*, 0 as level FROM workers WHERE workers.workerid = 5

UNION

SELECT workers.\*, level + 1 FROM workers

JOIN second\_worker ON (workers.workerid = second\_worker.mgr)

)

SELECT 'Nearest mgr is ' || people.firstname || ' ' || people.secondname || ' ID=' || workerid as "Result"

FROM first\_worker JOIN second\_worker USING (workerid)

JOIN people ON (first\_worker.pid = people.id)

ORDER BY first\_worker.level

LIMIT 1;

