



Бази даних та інформаційні системи

Тема 14. Агрегатні функції

СумДУ, каф КН.
2020

Агрегатні функції (Group Functions)

Агрегатна функція приймає в якості аргументу безліч рядків і повертає один рядок результату для кожної групи

EMP

EMPNO	ENAME	SAL
7839	KING	5000
7698	BLAKE	2850
7782	CLARK	1500
7566	JONES	2975
7654	MARTIN	1250
7499	ALLEN	1600
...		
7934	MILLER	1300

Яка максимальна
зарплата в
EMP?

MAX (SAL))

5000

Основні функції агрегування

Функція	Призначення	Припустимі аргументи
AVG	Середнє значення	Числа
COUNT	Кількість	Будь-який тип
MAX	Максимальне значення	Числа, дати, рядки
MIN	Мінімальне значення	Числа, дати, рядки
STDDEV	середньоквадратичне відхилення	Числа
SUM	Сума	Числа
VARIANCE	Дисперсія	Числа

Синтаксис

```
SELECT      group_function(column), ...
FROM        table
[WHERE      condition]
[ORDER BY   column]
```

- Приклад:

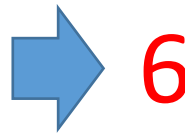
```
SELECT  AVG(sal) , MAX(sal) ,
        MIN(sal) , SUM(sal)
FROM    emp
WHERE   job LIKE '%SAL%'
```

Використання функції COUNT

- **COUNT** (*) повертає кількість рядків у результаті запиту:

1

```
SELECT COUNT (*)  
FROM emp  
WHERE deptno = 30
```



6

- **COUNT** (expr) повертає кількість **не порожніх** результатів:

2

```
SELECT COUNT (comm)  
FROM emp  
WHERE deptno = 30
```



4

EMPNO	ENAME	COMM	DEPTNO
7839	KING		10
7698	BLAKE		30
7782	CLARK		10
7566	JONES		20
7654	MARTIN	1400	30
7499	ALLEN	300	30
7844	TURNER	0	30
7900	JAMES		30
7521	WARD	500	30

Використання DISTINCT і COUNT

- **COUNT (DISTINCT expr)**
повертає кількість expr, які не є порожніми і не повторюються
- Скільки відділів, в яких є співробітники?

```
SELECT  
  COUNT(DISTINCT deptno)  
FROM    emp
```



3

EMPNO	ENAME	COMM	DEPTNO
7839	KING		10
7698	BLAKE		30
7782	CLARK		10
7566	JONES		20
7654	MARTIN	1400	30
7499	ALLEN	300	30
7844	TURNER	0	30
7900	JAMES		30
7521	WARD	500	30

Порожні значення (Null) і функції агрегування

- Агрегатні функції **пропускають Null** значення:

1

```
SELECT AVG (comm)  
FROM emp
```

550

- Якщо ви хочете їх включити - використовуйте функцію **NVL** :

2

```
SELECT AVG (NVL (comm, 0))  
FROM emp
```

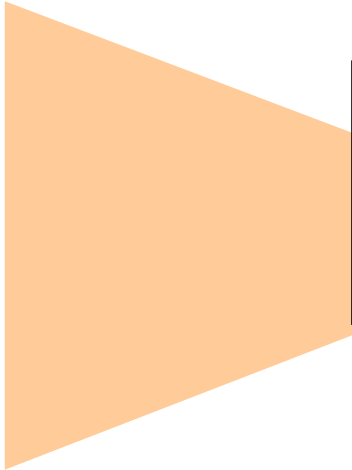


244,44

EMPNO	ENAME	COMM	DEPTNO
7839	KING		10
7698	BLAKE		30
7782	CLARK		10
7566	JONES		20
7654	MARTIN	1400	30
7499	ALLEN	300	30
7844	TURNER	0	30
7900	JAMES		30
7521	WARD	500	30

Групування інформації

EMPNO	ENAME	SAL	DEPTNO
7839	KING	5000	10
7698	BLAKE	2850	30
7782	CLARK	1500	10
7566	JONES	2975	20
7654	MARTIN	1250	30
7499	ALLEN	1600	30
7844	TURNER	1500	30




DEPTNO	avg (SAL)
10	3520
20	2975
30	1800

Синтаксис GROUP BY

- Можна розділити рядки на групи використовуючи специфікатор **GROUP BY**.

```
SELECT      column, group_function(column)
FROM        table
[WHERE      condition]
[GROUP BY group_by_expression]
[ORDER BY column]
```



Імена стовпців через кому.
Псевдоніми стовпців не допускаються
Псевдоніми таблиць допускаються

Використання GROUP BY

Всі стовпці у списку **SELECT**, для яких не використовується агрегатна функція, повинні бути вказані у специфікаторі **GROUP BY**.

```
SELECT deptno, AVG(sal)
FROM emp
GROUP BY deptno
```

DEPTNO	avg (SAL)
10	3520
20	2975
30	1800

Використання GROUP BY

- Стовпці, зазначені в **GROUP BY**, можуть не згадуватися в списку **SELECT**.

```
SELECT    AVG (sal)
FROM      emp
GROUP BY  deptno
```

Групування за кількома стовпцями

EMP

ENAME	JOB	SAL	DEPTNO
BLAKE	MANAGER	2850	30
CLARK	MANAGER	1500	10
JONES	MANAGER	2975	20
MARTIN	SALESMAN	1250	30
ALLEN	SALESMAN	1600	30
TURNER	SALESMAN	1500	30
JAMES	SALESMAN	950	10

Середня зарплата за посадами
працівників в різних відділах

DEPNO	JOB	avg (SAL)
30	MANAGER	2850
10	MANAGER	1500
20	MANAGER	2975
30	SALESMAN	1450
10	SALESMAN	950

```
SELECT deptno, job, AVG(sal)
FROM emp
GROUP BY deptno, job
ORDER BY job;
```

Помилки при складанні запитів

- Всі колонки з **Select**, до яких не застосовуються агрегатні функції повинні бути вказані в **GROUP BY**:

```
SELECT deptno, COUNT(ename)
FROM emp
```

ORA-00937: not a single-group group function
00937. 00000 - "not a single-group group function"

Необхідно додати **GROUP BY deptno**.


```
SELECT deptno, job, COUNT(ename)
FROM emp
GROUP BY deptno
```

ORA-00979: not a GROUP BY expression
00979. 00000 - "not a GROUP BY expression"

Потрібно додати колонку **job** в **GROUP BY**, або прибрати її з **SELECT**.

Обмеження на результати групування

- Вивести максимальну зарплату по відділах, в яких вона перевищує 8000



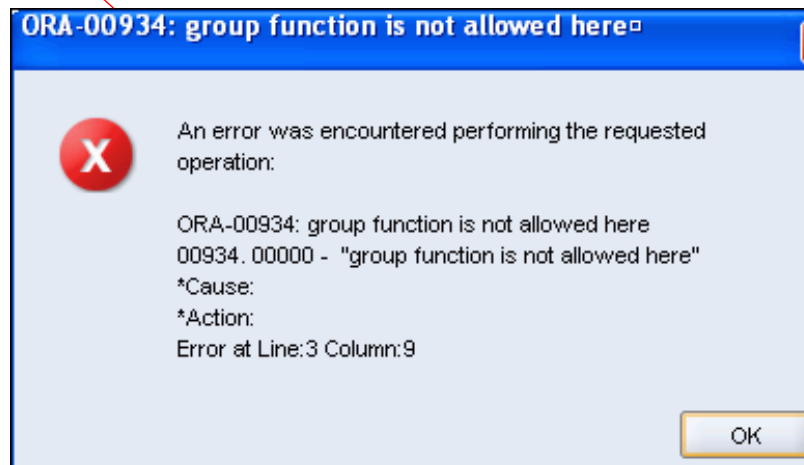
DEPTNO	SAL
10	4400
20	13000
20	6000
50	5800
50	2500
50	2600
50	3100
50	3500
60	4200
60	6000
...	
110	12000

DEPTNO	max (SAL)
20	13000
110	12000

Помилки при складанні запитів

- **Не можна** використовувати функції агрегування в **WHERE**.
- Для цього є специфікатор **HAVING**.

```
SELECT    deptno, AVG(sal)
FROM      emp
WHERE     AVG(sal) > 8000
GROUP BY  deptno;
```



Обмеження з використанням HAVING

При використанні **HAVING** сервер БД виконує дії в наступному порядку:

1. Рядки групуються;
2. Застосовується агрегатна функція;
3. Повертаються групи, які задовольняють умовам в **HAVING**.

```
SELECT      column, group_function  
FROM        table  
[WHERE      condition]  
[GROUP BY   group_by_expression]  
[HAVING     group_condition]  
[ORDER BY   column]
```

Використання HAVING

- Вивести максимальну зарплату по відділах, де вона перевищує 8000:

```
SELECT deptno, MAX(sal)
FROM emp
GROUP BY deptno
HAVING MAX(sal) > 8000
```

DEPTNO	max (SAL)
20	13000
110	12000

Використання HAVING і WHERE

```
SELECT    deptno, SUM(sal)
FROM      emp
WHERE     deptno != 20
GROUP BY  deptno
HAVING    SUM(sal) > 10000
ORDER BY  SUM(sal)
```

DEPTNO	sum (SAL)
30	12000
10	18000

Використання HAVING і WHERE

```
SELECT deptno, SUM(sal)
FROM emp
WHERE deptno != 20 AND job != 'PRESIDENT'
GROUP BY deptno
HAVING SUM(sal) > 10000
ORDER BY SUM(sal)
```

DEPTNO	sum(SAL)
30	18000
10	12000

Вкладені агрегатні функції

- Отримання максимальної середньої зарплати по відділах:

```
SELECT  MAX (AVG (sal))  
FROM    emp  
GROUP BY deptno
```

Питання:

Які з тверджень про **GROUP BY** правильні?

1. Не можна використовувати псевдоніми (**alias**) атрибутів в реченні **GROUP BY**.
2. Всі стовпчики зазначені в **GROUP BY** повинні бути вказані в **SELECT**.
3. Умова **WHERE** застосовується до рядків до групування.
4. Умова **HAVING** застосовується до рядків після групування.
5. Специфікатор **GROUP BY** задає порядок сортування рядків.

Підсумки

Сьогодні ми розглянули:

- Агрегатні функції **COUNT**, **MAX**, **MIN**, **SUM**, **AVG** ;
- Як записувати запити з використанням **GROUP BY** ;
- Як записувати запити з використанням **HAVING** .

```
SELECT      column, group_function  
FROM        table  
[WHERE      condition]  
[GROUP BY group_by_expression]  
[HAVING     group_condition]  
[ORDER BY  column]
```


- Зведена таблиця (Pivot Table)
- Розширені можливості групування
- Аналітичні функції
- Задання вікна аналітичної функції

Зведена таблиця (Pivot Table)

Кількість співробітників у відділах за посадами:

```
SELECT * FROM
  ( SELECT dname, job
    FROM emp e? fept d
    WHERE e.deptno = d.deptno)
PIVOT
(COUNT(job) FOR job IN ('MANAGER', 'SALESMAN',
                           'CLERK', 'ANALYST')) pvt ;
```

DNAME	'MANAGER'	'SALESMAN'	'CLERK'	'ANALYST'
ACCOUNTING	1	0	1	0
RESEARCH	1	0	2	2
SALES	1	4	1	0
SALES2	0	3	0	0

Зведена таблиця (Pivot Table)

```
SELECT * FROM
(
  SELECT column1, column2
  FROM tables
  WHERE conditions
)
PIVOT
(
  aggregate_function(column2)
  FOR column2
  IN ( expr1, expr2, ... expr_n) | subquery
)
ORDER BY expression [ ASC | DESC ];
```

IN (expr1, expr2, ... expr_n) - список значень для повороту column2 у заголовку крос-табличного результату запиту

Проміжні підсумки

```
select deptno, job, count(empno) headcount  
from emp  
group by ROLLUP(deptno, job);
```

```
select deptno, job, count(empno) headcount  
from emp  
group by CUBE(deptno, job);
```

GROUP BY ROLLUP

```
SQL> select deptno, job,  
2         count(empno) headcount  
3 from emp  
4 group by ROLLUP(deptno,job);
```

DEPTNO	JOB	HEADCOUNT

10	CLERK	1
10	MANAGER	1
10	PRESIDENT	1
10		3
20	CLERK	2
20	ANALYST	2
20	MANAGER	1
20		5
30	CLERK	1
30	MANAGER	1
30	SALESMAN	4
30		6
		14

12 rows selected.

GROUP BY CUBE

```
SQL> select deptno, job,  
2         count(empno) headcount  
3 from employees  
4 group by CUBE(deptno, job);
```

Обчислює всі можливі варіанти
проміжних підсумкових значень в
операції GROUP BY

DEPTNO	JOB	HEADCOUNT

		14
	CLERK	4
	ANALYST	2
	MANAGER	3
	SALESMAN	7
	PRESIDENT	1
10		3
10	CLERK	1
10	MANAGER	1
10	PRESIDENT	1
20		5
20	CLERK	2
20	ANALYST	2
20	MANAGER	1
30		6
30	CLERK	1
30	MANAGER	1
30	SALESMAN	4

18 rows selected.

Проміжні підсумки

Rollup

- **ROLLUP (DEP, JOB):**
Group by DEP, JOB
Union
Group by DEP
union
()

Cube

- **CUBE (DEP, JOB)**
Group by DEP, JOB
Union
Group by DEP
Union
Group by JOB
Union
()


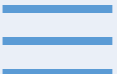

Grouping sets

```
SELECT loc, job, SUM(sal)
FROM emp, dept
WHERE emp.deptno = dept.deptno
GROUP BY
GROUPING SETS ((loc,job),(job))
```

LOC	JOB	SUM(SAL)
-----	-----	-----
DALLAS	CLERK	1900
CHICAGO	CLERK	950
NEW_YORK	CLERK	1300
	CLERK	4150
DALLAS	ANALYST	6000
	ANALYST	6000
DALLAS	MANAGER	2975
CHICAGO	MANAGER	2850
NEW_YORK	MANAGER	1500
	MANAGER	7325
CHICAGO	SALESMAN	5600
HONKONG	SALESMAN	7350
	SALESMAN	12950
NEW_YORK	PRESIDENT	5000
	PRESIDENT	5000

15 rows selected.

Розставлення дужок в GROUPING SETS

GROUP BY GROUPING SETS ((loc, deptno, job))		GROUP BY loc, deptno, job
GROUP BY GROUPING SETS (loc, deptno, job)		GROUP BY loc UNION GROUP BY deptno UNION GROUP BY job
GROUP BY GROUPING SETS (loc, (deptno, job))		GROUP BY loc UNION GROUP BY deptno, job

GROUPING, GROUPING_ID

- Функції **GROUPING**, **GROUPING_ID** дозволяють визначити, чи є рядок результатом групування

	Аргумент	
GROUPING (expr)	Стовпчик або вираз з GROUP BY	1 - якщо рядок згруповано по аргументу Інакше - 0.
GROUPING_ID (...,expr3, expr2,expr1)		0 - якщо рядок не групували 1 - якщо рядок групували по expr1 2 - якщо рядок групували по expr2 3 - якщо рядок групували по expr1і expr2 4 - якщо рядок групували по expr3....

Grouping

```
SQL> select deptno,  
2 case GROUPING(job)  
3     when 0 then job  
4     when 1 then '**total**'  
5 end job,  
6 count(empno) headcount  
7 from emp  
8 group by  
9 rollup(deptno, job);
```

DEPTNO	JOB	HEADCOUNT
-----	-----	-----
10	CLERK	1
10	MANAGER	1
10	PRESIDENT	1
10	**total**	3
20	CLERK	2
20	ANALYST	2
20	MANAGER	1
20	**total**	5
30	CLERK	1
30	MANAGER	1
30	SALESMAN	4
30	**total**	6
60	SALESMAN	3
60	**total**	3
	total	17

15 rows selected.

Grouping_ID

```
select deptno,  
       case GROUPING_ID(deptno, job)  
       when 0 then job  
       when 1 then '**dept **'  
       when 3 then '**total**'  
       end job  
 , count(empno) headcount  
from emp  
group by rollup(deptno, job);
```

DEPTNO	JOB	HEADCOUNT
10	CLERK	1
10	MANAGER	1
10	PRESIDENT	1
10	**dept **	3
20	CLERK	2
20	ANALYST	2
20	MANAGER	1
20	**dept **	5
30	CLERK	1
30	MANAGER	1
30	SALESMAN	4
30	**dept **	6
60	SALESMAN	3
60	**dept **	3
	total	17

Аналітичні функції

- Розраховують ранги і відсотки в результуючому наборі;
- Агрегують значення в межах виділеного вікна;
- Здійснюють пошук значення по зміщенню від поточного рядка;
- Знаходять перші і останні значення в виділеній групі.

```
SELECT analytical-function(col-expr)  
OVER (window-spec) [AS col-alias]  
...  
FROM table-name
```

Аналітичні функції

Аналітика, важко відображається засобами стандартного SQL.

Приклади задач:

- Підрахунок підсумків, що нарастають (показати підсумки, що нарастають по зарплаті по рядках для кожного співробітника);
- Підрахунок відсотків в групі (який відсоток від загальної зарплати становить зарплата кожного співробітника);
- Вибірка перших N співробітників з найбільшими зарплатами;
- Підрахунок змінного середнього (отримати середнє значення по попередніх N рядках);
- Виконання ранжування (показати ранг зарплати співробітника серед інших співробітників).

DENSE_RANK, RANK

- Виконайте ранжування службовців за зарплатою в межах кожного з підрозділів

```
SELECT deptno, ename,  
RANK() OVER (PARTITION BY deptno ORDER BY sal DESC) AS n_rank  
FROM emp  
ORDER BY deptno, n_rank;
```

```
SELECT deptno, ename,  
dense_rank() OVER (PARTITION BY deptno ORDER BY sal DESC) AS  
n_rank  
FROM emp  
ORDER BY deptno, n_rank;
```

RANK

DEPTNO	ENAME	N_RANK

10	KING	1
10	CLARK	2
10	MILLER	3
20	FORD	1
20	SCOTT	1
20	JONES	3
20	ADAMS	4
20	SMITH	5
30	BLAKE	1
30	ALLEN	2
30	TURNER	3
30	MARTIN	4
30	WARD	4
30	JAMES	6
60	JET LI	1
60	BRUCE LEE	2
60	JACKIE CHAN	3

17 rows selected.

DENSE_RANK

DEPTNO	ENAME	N_RANK

10	KING	1
10	CLARK	2
10	MILLER	3
20	FORD	1
20	SCOTT	1
20	JONES	2
20	ADAMS	3
20	SMITH	4
30	BLAKE	1
30	ALLEN	2
30	TURNER	3
30	MARTIN	4
30	WARD	4
30	JAMES	5
60	JET LI	1
60	BRUCE LEE	2
60	JACKIE CHAN	3

17 rows selected.

Ще один приклад з Rank

```
SELECT ename,  
rank() OVER (ORDER BY sal DESC) n_rank_desc,  
rank() OVER (ORDER BY sal ASC) n_rank_asc,  
rank() OVER (ORDER BY deptno ASC) d_rank_asc  
FROM emp  
ORDER BY n_rank_desc
```

Ще один приклад з Rank

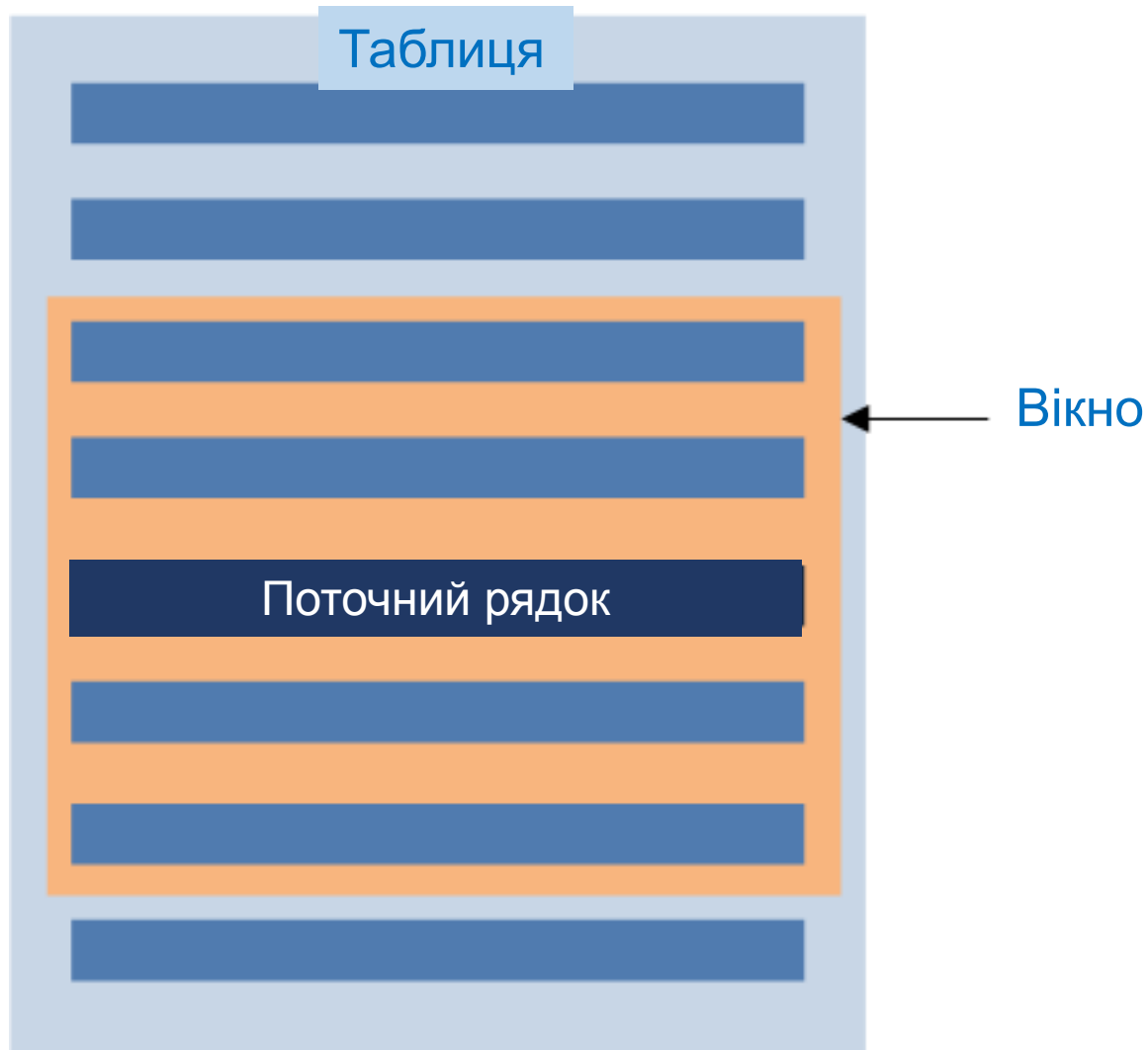
ENAME	N_RANK_DESC	N_RANK_ASC	D_RANK_ASC
KING	1	18	1
FORD	2	16	4
SCOTT	2	16	4
JONES	4	15	4
BLAKE	5	14	9
JET LI	6	13	15
DR NO	7	11	18
BRUCE LEE	7	11	15
JACKIE CHAN	9	10	15
ALLEN	10	9	9
TURNER	11	7	9
CLARK	11	7	1
MILLER	13	6	1
WARD	14	4	9
MARTIN	14	4	9
ADAMS	16	3	4
JAMES	17	2	9
SMITH	18	1	4

18 rows selected.

Параметри вікна

```
SELECT deptno, ename,  
RANK() OVER (PARTITION BY deptno ORDER BY sal DESC) AS n_rank  
FROM emp  
ORDER BY deptno, n_rank;
```

Аналітичні функції



Способи завдання вікна

- PARTITION BY expr
- ROWS UNBOUNDED PRECEDING
- ROWS UNBOUNDED PRECEDING AND CURRENT ROW
- ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
- RANGE BETWEEN 10 PRECEDING AND CURRENT ROW
- RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
- RANGE BETWEEN CURRENT ROW AND CURRENT ROW

PART and ORDERED QUERY

Знайти 5 осіб з найвищою зарплатою

- Спроба №1(Невдала)

```
1 SELECT e.ename, e.sal, rownum rn
2 FROM emp e
3 WHERE ROWNUM <= 5
4 ORDER BY sal DESC
```

ENAME	SAL	RN
-----	-----	-----
KING	5000	1
JONES	2975	4
BLAKE	2850	2
CLARK	1500	3
MARTIN	1250	5

Проблема: де працівники з зарплатою в 3000?

Знайти 5 осіб з найвищою зарплатою

- Спроба №2 (те що треба!)

```
1  SELECT * FROM (  
2      SELECT e.ename, e.sal  
3      FROM emp e  
4      ORDER BY sal DESC  
5  )  
6  WHERE ROWNUM <= 5
```

ENAME	SAL

KING	5000
FORD	3000
SCOTT	3000
JONES	2975
BLAKE	2850

Розбивка на сторінки (pagination)



Перші 5:

```
select * from  
    (select * from emp order by sal DESC)  
where rownum <= 5
```

Інші (неправильно):

```
select * from  
    (select * from emp order by sal DESC)  
where rownum >= 5
```

Проблема:

No rows selected

Правильна пагінація

- <http://www.oracle.com/technetwork/issue-archive/2006/06-sep/o56asktom-086197.html>

```
1 Select  * from (
2   Select /*+ FIRST_ROWS(n) */ a.*, ROWNUM rnum from(
3     Select * from emp order by sal DESC
2   ) a
2   where ROWNUM <= :START )
1 where rnum  >= :END;
```

Спосіб 2: Аналітична функція ROW_NUMBER

Перші 5:

```
1 SELECT  *
2 FROM    (
3     SELECT  e.name, e.sal,
4             row_number() OVER (ORDER BY SAL DESC) rn
5     FROM    emp e
6 )
7 WHERE rn <= 5;
```

C :start до :end :

```
1 SELECT  *
2 FROM    (
3     SELECT  e.name, e.sal,
4             row_number() OVER (ORDER BY SAL DESC) rn
5     FROM    emp e
6 )
7 WHERE rn BETWEEN :start and :end;
```

DISTINCT на частину полів:

Приклад - по одному представнику кожної посади

Варіант 1

```
SELECT  max(ename) , JOB
FROM    EMP
GROUP BY JOB;
```

Варіант 2

```
SELECT *
FROM    EMP
WHERE   ROWID IN (
                SELECT MAX(ROWID) FROM EMP GROUP BY JOB
            )
```

Варіант 3

```
SELECT  DISTINCT
        FIRST_VALUE(ename) OVER(PARTITION BY job ORDER BY ename) ,
        JOB
FROM    EMP;
```