Group BY

&

Window FUNCTION

Table Create

```sql
CREATE DATABASE Group_Window_HK;
USE Group_Window_HK;

CREATE TABLE Employee(
    id INT IDENTITY(1,1)
        CONSTRAINT PK_Employee_id PRIMARY KEY,

    fname varchar(10),
    dept varchar(10),
    salary INT
);
```

```
SELECT * FROM Employee;
```

	id	fname	dept	salary
1	1	AA	IT	40000
2	2	BB	HR	30000
3	3	CC	IT	50000
4	4	DD	IT	60000
5	5	EE	HR	40000
6	6	FF	HR	50000
7	7	GG	IT	40000
8	8	HH	HR	30000
9	9	II	IT	40000
10	10	JJ	HR	30000
11	11	KK	IT	50000

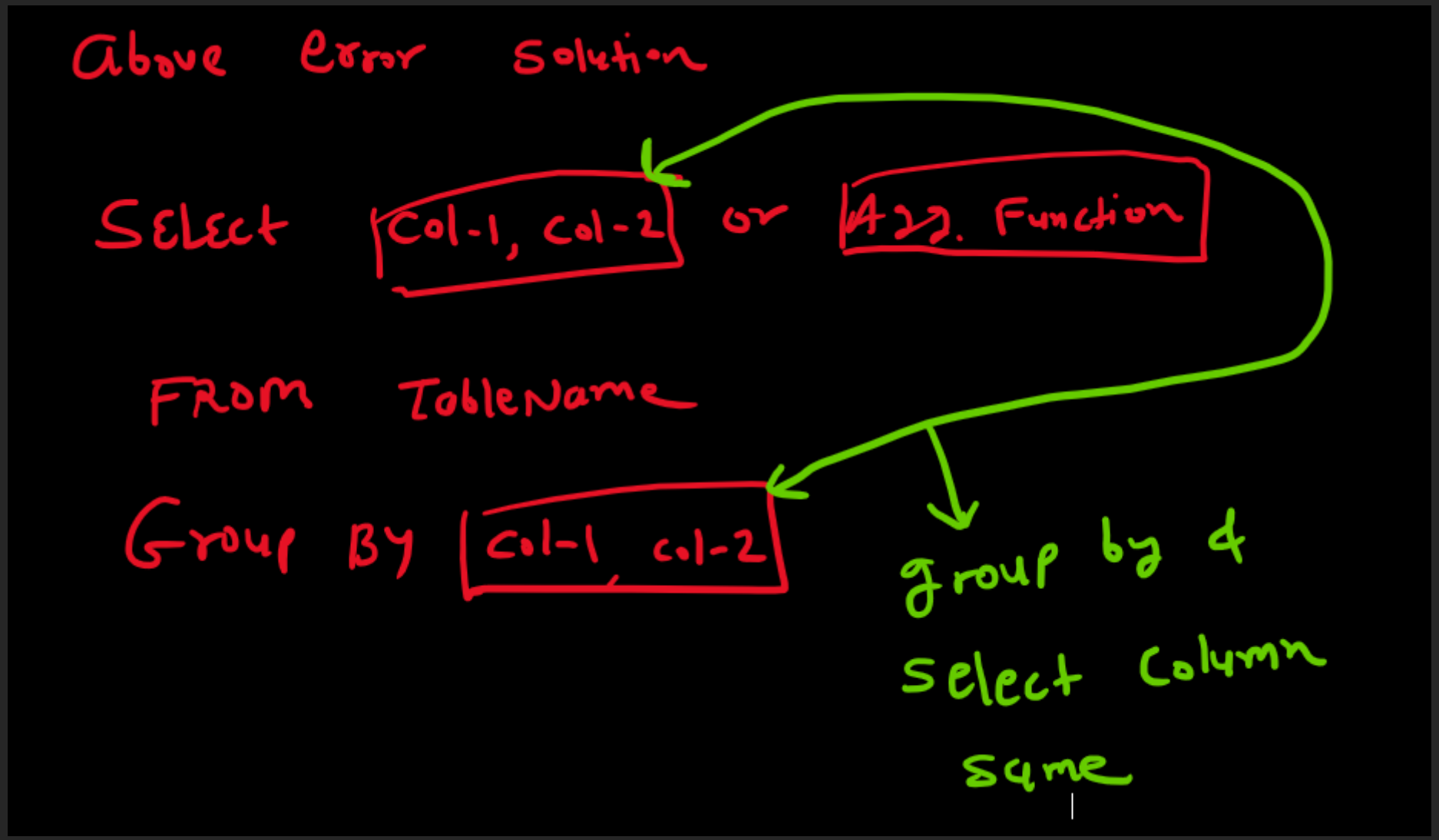
Group BY

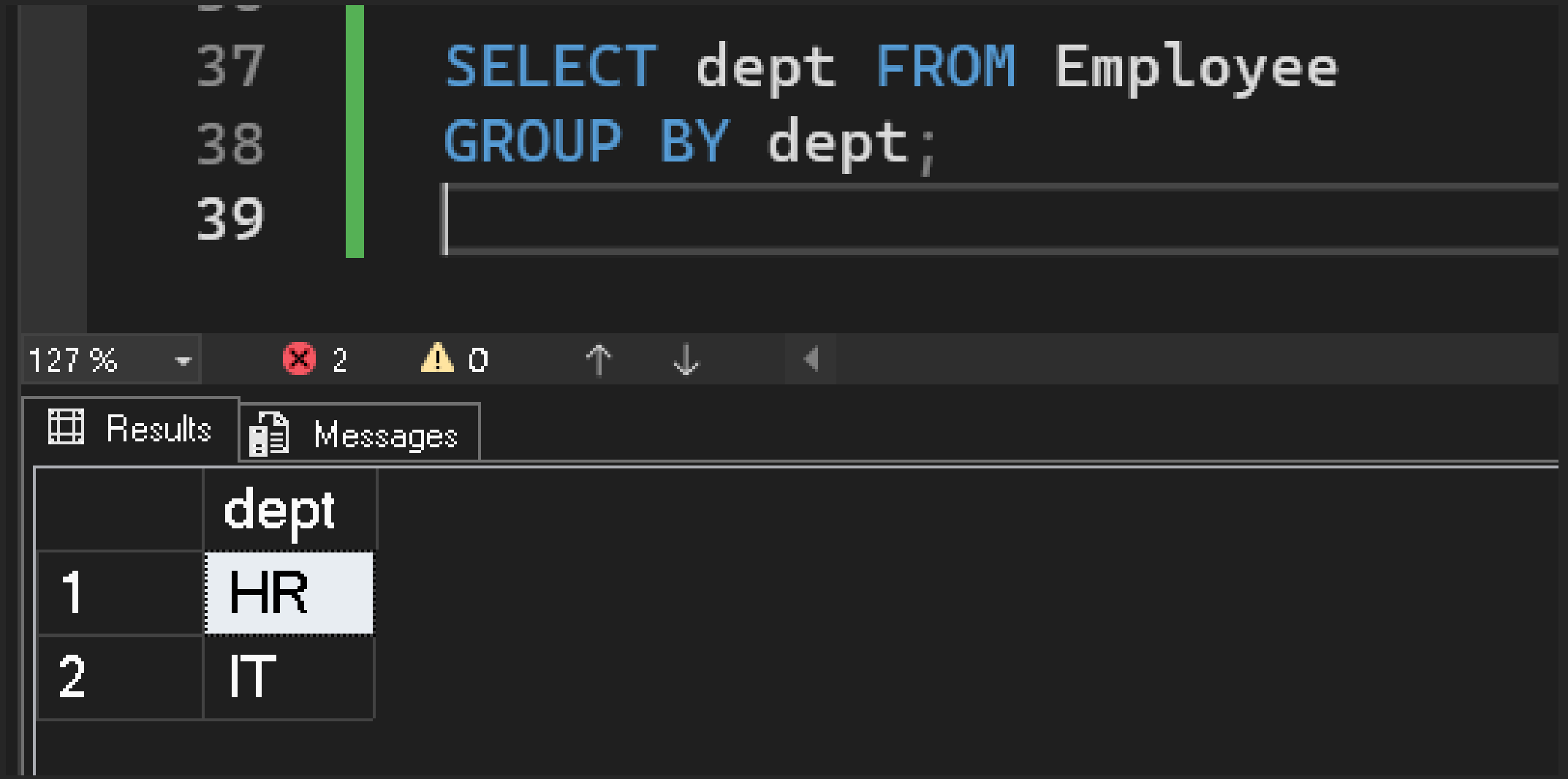
```sql
--GROUP BY

SELECT * FROM Employee
GROUP BY dept;
```

127 % 2 0 Ln: 33 Ch:

Messages

```
Msg 8120, Level 16, State 1, Line 33
Column 'Employee.id' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

Completion time: 2025-11-04T22:31:45.5613068+05:30
```


Above error solution
Select
Col-1, col-2
or
Agg. Function
FROM TableName
Group BY
col-1 col-2
group by &
select Column
same


37 SELECT dept FROM Employee
38 GROUP BY dept;
39
127 %
2
0
Results
Messages
dept
1 HR
2 IT

```
SELECT fname,dept,salary FROM Employee ORDER BY Dept,salary
```

	fname	dept	salary
1	BB	HR	30000
2	HH	HR	30000
3	JJ	HR	30000
4	EE	HR	40000
5	FF	HR	50000
6	GG	IT	40000
7	II	IT	40000
8	AA	IT	40000
9	KK	IT	50000
10	CC	IT	50000
11	DD	IT	60000

```sql
SELECT
        dept
       ,MIN(salary) AS 'Min Salary'
       ,MAX(salary) AS 'Max Salary'
       ,AVG(salary) AS 'Avg Salary'
       ,SUM(salary) AS 'Total Salary'
FROM Employee
GROUP BY dept
```

27 % 1 0

Results Messages

	dept	Min Salary	Max Salary	Avg Salary	Total Salary
1	HR	30000	50000	36000	180000
2	IT	40000	60000	46666	280000

Step-1

```
SELECT
      dept
     ,MIN(salary) AS 'Min Salary'
     ,MAX(salary) AS 'Max Salary'
     ,AVG(salary) AS 'Avg Salary'
     ,SUM(salary) AS 'Total Salary'
FROM Employee
GROUP BY dept
```

HR, IT

Group-1

Group-2

	id	fname	dept	salary
1	2	BB	HR	30000
2	8	HH	HR	30000
3	10	JJ	HR	30000
4	5	EE	HR	40000
5	6	FF	HR	50000
6	7	GG	IT	40000
7	9	II	IT	40000
8	1	AA	IT	40000
9	11	KK	IT	50000
10	3	CC	IT	50000
11	4	DD	IT	60000

Step-2

SELECT Column Pick Group-1 & Group-2

```
SELECT
      dept
     ,MIN(salary) AS 'Min Salary'
     ,MAX(salary) AS 'Max Salary'
     ,AVG(salary) AS 'Avg Salary'
     ,SUM(salary) AS 'Total Salary'
FROM Employee
GROUP BY dept
```

HR, IT

dept

min mux Avg Sum

Group-1

Group-2

	id	fname	dept	salary
1	2	BB	HR	30000
2	8	HH	HR	30000
3	10	JJ	HR	30000
4	5	EE	HR	40000
5	6	FF	HR	50000
6	7	GG	IT	40000
7	9	II	IT	40000
8	1	AA	IT	40000
9	11	KK	IT	50000
10	3	CC	IT	50000
11	4	DD	IT	60000

```sql
SELECT
       dept
      ,MIN(salary) AS 'Min Salary'
      ,MAX(salary) AS 'Max Salary'
      ,AVG(salary) AS 'Avg Salary'
      ,SUM(salary) AS 'Total Salary'
      ,COUNT(*) AS 'Number of Employee this department'
FROM Employee
GROUP BY dept
```

	dept	Min Salary	Max Salary	Avg Salary	Total Salary	Number of Employee this department
1	HR	30000	50000	36000	180000	5
2	IT	40000	60000	46666	280000	6

Having

```sql
SELECT
       dept
      ,MIN(salary) AS 'Min Salary'
      ,MAX(salary) AS 'Max Salary'
      ,AVG(salary) AS 'Avg Salary'
      ,SUM(salary) AS 'Total Salary'
      ,COUNT(*) AS 'Number of Employee this department'
FROM Employee
GROUP BY dept
HAVING MAX(salary) > 50000
```

	dept	Min Salary	Max Salary	Avg Salary	Total Salary	Number of Employee this department
1	IT	40000	60000	46666	280000	6

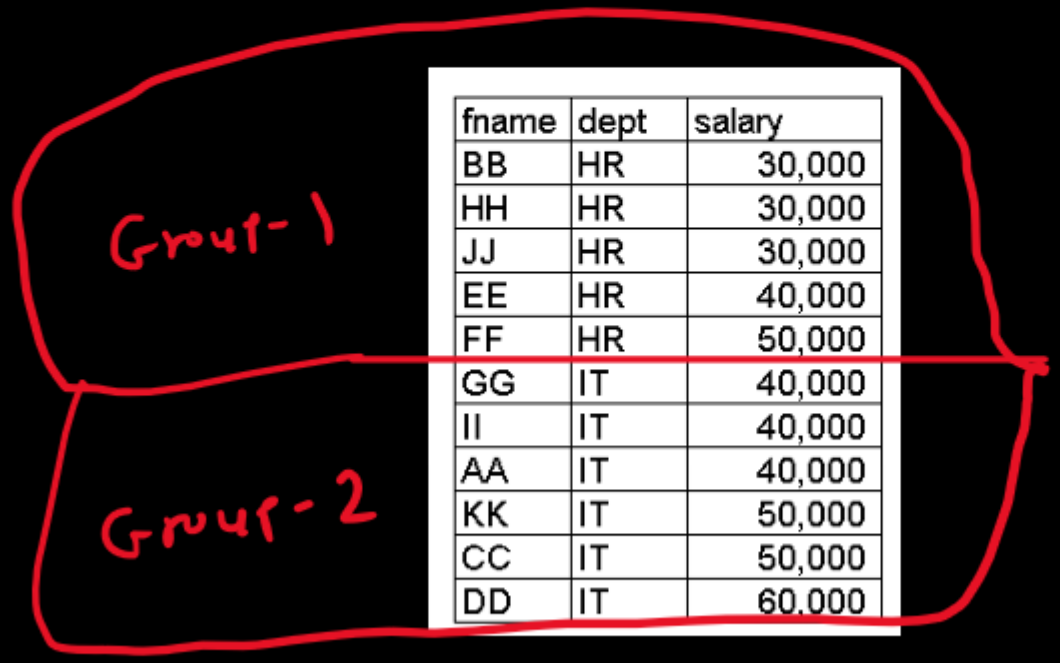
```sql
SELECT
      dept
     ,MIN(salary) AS 'Min Salary'
     ,MAX(salary) AS 'Max Salary'
     ,AVG(salary) AS 'Avg Salary'
     ,SUM(salary) AS 'Total Salary'
     ,COUNT(*) AS 'Number of Employee this department'
FROM Employee
GROUP BY dept
HAVING MAX(salary) > 50000
```

Step-1 → Group By

Step-2 → see Select column

Step-3 → Group filter based on Having Condition

Step-1 Group By dept



fname	dept	salary
BB	HR	30,000
HH	HR	30,000
JJ	HR	30,000
EE	HR	40,000
FF	HR	50,000
GG	IT	40,000
II	IT	40,000
AA	IT	40,000
KK	IT	50,000
CC	IT	50,000
DD	IT	60,000

Step-2

Results | Messages

	dept	Min Salary	Max Salary	Avg Salary	Total Salary	Number of Employee this department
1	HR	30000	50000	36000	180000	5
2	IT	40000	60000	46666	280000	6

Step-3 Having MAX(salary) > 50,000

	dept	Min Salary	Max Salary	Avg Salary	Total Salary	Number of Employee this department
1	IT	40000	60000	46666	280000	6

OVER()

[Aggregat Function or Ranking Function or Analytic Functions]
OVER(
PARTITION BY ColumnName
ORDER BY ColumnName ASC/DESC
)

- PARTITION BY --> optional
- ORDER BY --> Optional for Aggregate Functions
- ORDER BY --> Required for Ranking / Analytic Window Functions

- The OVER clause allows you to perform aggregate calculations without grouping the rows.

```
function_name (expression)
OVER (
   [PARTITION BY column_list]
   [ORDER BY column_list]
   [ROWS/RANGE BETWEEN ...]
)
```

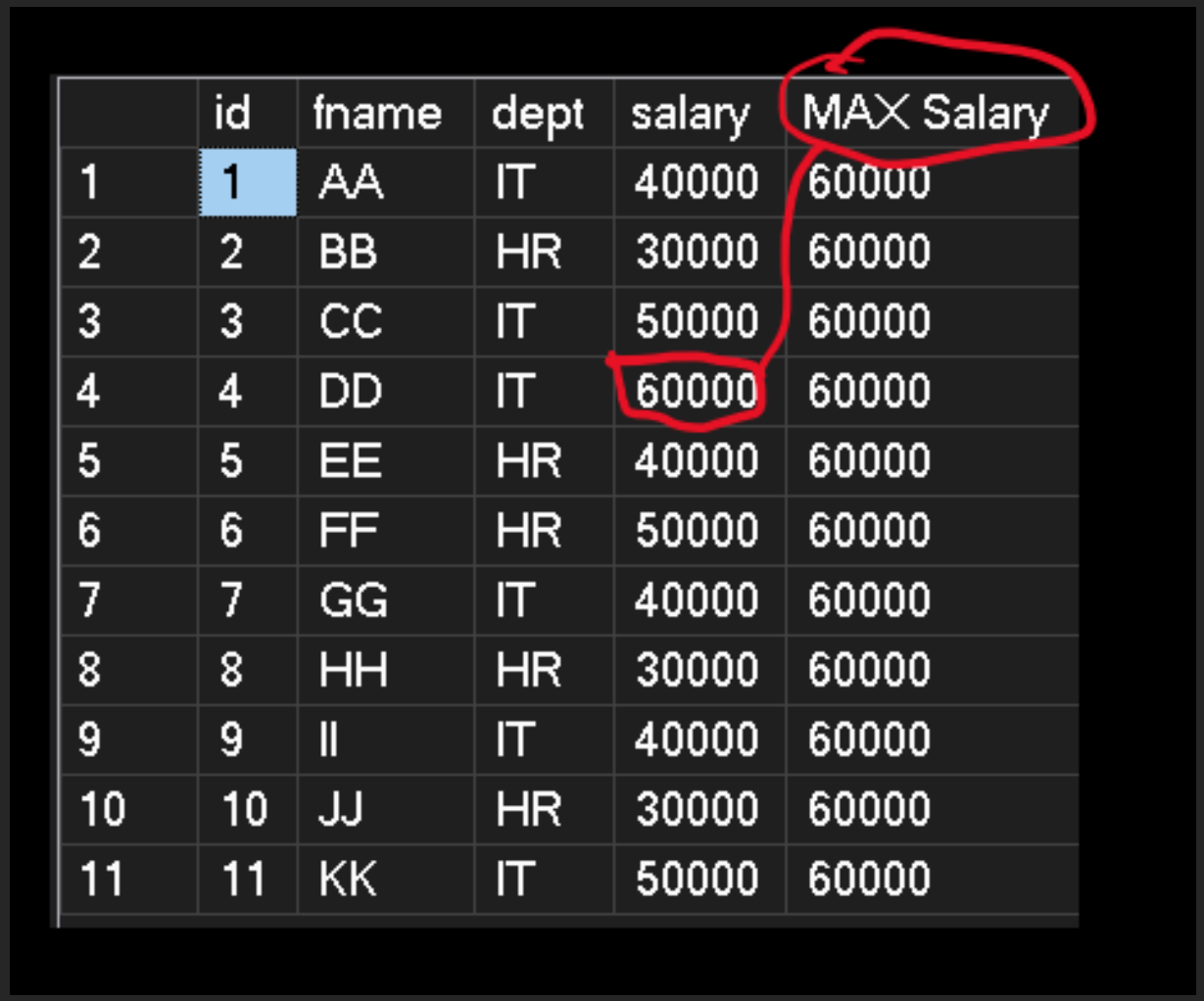
- function_name → Aggregate (e.g., SUM, AVG) or ranking function (e.g., ROW_NUMBER).
- PARTITION BY → Splits data into groups (like GROUP BY but rows remain).
- ORDER BY → Defines order inside each partition.
- ROWS / RANGE → Defines frame of rows for calculation.

```sql
--Part: 6

--issue
--all column with max salary find
SELECT *,MAX(salary) FROM Employee
```

```
Msg 8120, Level 16, State 1, Line 80
Column 'Employee.id' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.
```

- this problem solution use over function

```sql
--all column with max salary find using over function
SELECT *,

MAX(salary) OVER() AS 'MAX Salary'

FROM Employee
```



	id	fname	dept	salary	MAX Salary
1	1	AA	IT	40000	60000
2	2	BB	HR	30000	60000
3	3	CC	IT	50000	60000
4	4	DD	IT	60000	60000
5	5	EE	HR	40000	60000
6	6	FF	HR	50000	60000
7	7	GG	IT	40000	60000
8	8	HH	HR	30000	60000
9	9	II	IT	40000	60000
10	10	JJ	HR	30000	60000
11	11	KK	IT	50000	60000

```sql
--ORDER BY (SELECT NULL) using
SELECT *,

MAX(salary) OVER(ORDER BY (SELECT NULL)) 'MAX Salary'

FROM Employee
```

id	fname	dept	salary	MAX Salary
1	AA	IT	40000	60000
2	BB	HR	30000	60000
3	CC	IT	50000	60000
4	DD	IT	60000	60000
5	EE	HR	40000	60000
6	FF	HR	50000	60000
7	GG	IT	40000	60000
8	HH	HR	30000	60000
9	II	IT	40000	60000
10	JJ	HR	30000	60000
11	KK	IT	50000	60000

using below both method order by not required

Method-1: OVER()
Method-2: OVER(ORDER BY (SELECT NULL))

```sql
--all column with running total find using over function based on employee id
SELECT *,

SUM(salary) OVER(ORDER BY id) 'Running Total'

FROM Employee
```

```sql
SELECT *,
SUM(salary) OVER(ORDER BY id) 'Running Total'
FROM Employee
```

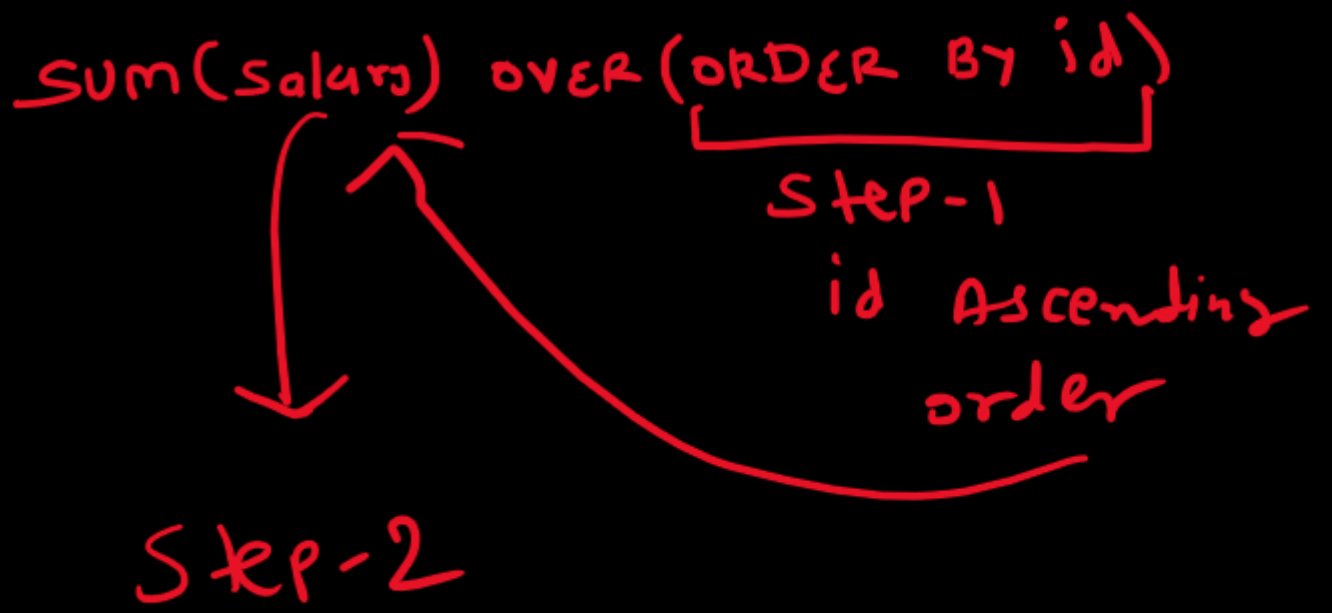
id	fname	dept	salary	Running Total
1	AA	IT	40000	40000
2	BB	HR	30000	70000
3	CC	IT	50000	120000
4	DD	IT	60000	180000
5	EE	HR	40000	220000
6	FF	HR	50000	270000
7	GG	IT	40000	310000
8	HH	HR	30000	340000
9	II	IT	40000	380000
10	JJ	HR	30000	410000
11	KK	IT	50000	460000

SUM(salary) OVER(ORDER BY id)

Step-1
id Ascending order

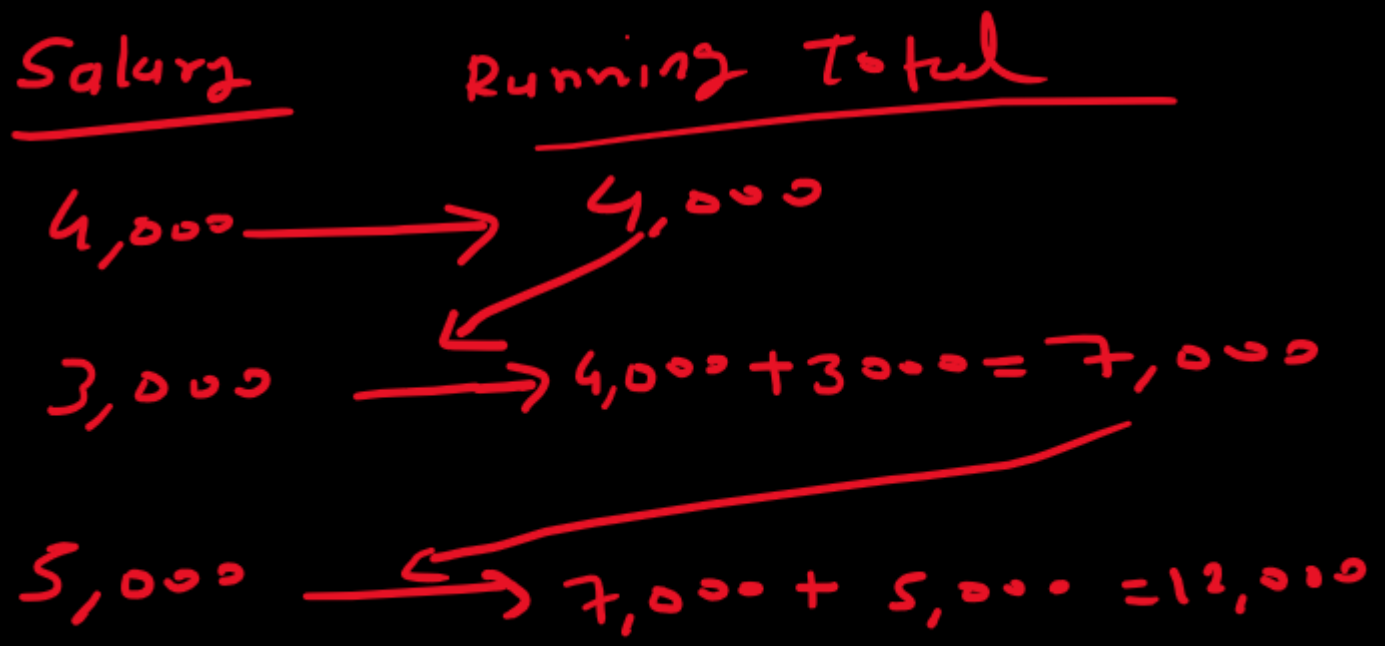
```
SELECT *,
SUM(salary) OVER(ORDER BY id) 'Running Total'
FROM Employee
```

id	fname	dept	salary	Running Total
1	AA	IT	40000	40000
2	BB	HR	30000	70000
3	CC	IT	50000	120000
4	DD	IT	60000	180000
5	EE	HR	40000	220000
6	FF	HR	50000	270000
7	GG	IT	40000	310000
8	HH	HR	30000	340000
9	II	IT	40000	380000
10	JJ	HR	30000	410000
11	KK	IT	50000	460000



```sql
SELECT *,
SUM(salary) OVER(ORDER BY id) 'Running Total'
FROM Employee
```

id	fname	dept	salary	Running Total
1	AA	IT	40000	40000
2	BB	HR	30000	70000
3	CC	IT	50000	120000
4	DD	IT	60000	180000
5	EE	HR	40000	220000
6	FF	HR	50000	270000
7	GG	IT	40000	310000
8	HH	HR	30000	340000
9	II	IT	40000	380000
10	JJ	HR	30000	410000
11	KK	IT	50000	460000



Salary	ROW_NUMBER	Rank	DENSE_RANK
10000	1	1	1
9000	2	2	2
9000	3	2	2
8000	4	4	3

ROW_NUMBER() OVER(ORDER BY Salary DESC)
ROW_NUMBER() → Always assigns a unique sequential number (1, 2, 3, 4).

RANK() OVER(ORDER BY Salary DESC)
RANK() → Same rank for ties (two 9000s get rank 2), and **next rank skips** to 4.

DENSE_RANK() OVER(ORDER BY Salary DESC)
DENSE_RANK() → Same rank for ties, but **no gaps** (next rank is 3).

Salary	ROW_NUMBER	Rank	DENSE_RANK
10000	1	1	1
9000	2	2	2
9000	3	2	2
8000	4	4	3

In short Row Number, Rank, & DENSE RANK Same Partition By, same ORDER By condition so 1st Row Number find

Salary	ROW_NUMBER	Rank	DENSE_RANK
10000	1	1	1
9000	2	2	2
9000	3	2	2
8000	4	4	3

```sql
--ROW_NUMBER
SELECT *,
    ROW_NUMBER()
    OVER(
        ORDER BY (SELECT NULL)
    ) AS 'ROW_NUMBER'
FROM Employee;
```

id	fname	dept	salary	ROW_NUMBER
1	AA	IT	40000	1
2	BB	HR	30000	2
3	CC	IT	50000	3
4	DD	IT	60000	4
5	EE	HR	40000	5
6	FF	HR	50000	6
7	GG	IT	40000	7
8	HH	HR	30000	8
9	II	IT	40000	9
10	JJ	HR	30000	10
11	KK	IT	50000	11

```sql
SELECT *,
    ROW_NUMBER()
    OVER(
        ORDER BY dept ASC
    )AS 'ROW_NUMBER'
FROM Employee;
```

id	fname	dept	salary	ROW_NUMBER
2	BB	HR	30000	1
5	EE	HR	40000	2
6	FF	HR	50000	3
8	HH	HR	30000	4
10	JJ	HR	30000	5
11	KK	IT	50000	6
9	II	IT	40000	7
7	GG	IT	40000	8
3	CC	IT	50000	9
4	DD	IT	60000	10
1	AA	IT	40000	11

```sql
SELECT *,
    ROW_NUMBER()
    OVER(
        PARTITION BY dept
        ORDER BY fname DESC
    ) AS 'ROW_NUMBER'
FROM Employee;
```

id	fname	dept	salary	ROW_NUMBER
10	JJ	HR	30000	1
8	HH	HR	30000	2
6	FF	HR	50000	3
5	EE	HR	40000	4
2	BB	HR	30000	5
11	KK	IT	50000	1
9	II	IT	40000	2
7	GG	IT	40000	3
4	DD	IT	60000	4
3	CC	IT	50000	5
1	AA	IT	40000	6

```sql
SELECT * FROM Employee;
```

id	fname	dept	salary
1	AA	IT	40000
2	BB	HR	30000
3	CC	IT	50000
4	DD	IT	60000
5	EE	HR	40000
6	FF	HR	50000
7	GG	IT	40000
8	HH	HR	30000
9	II	IT	40000
10	JJ	HR	30000
11	KK	IT	50000

```sql
SELECT *,
    ROW_NUMBER()
    OVER(
        PARTITION BY dept
        ORDER BY fname DESC
    ) AS 'ROW_NUMBER'
FROM Employee;
```

id	fname	dept	salary	ROW_NUMBER
10	JJ	HR	30000	1
8	HH	HR	30000	2
6	FF	HR	50000	3
5	EE	HR	40000	4
2	BB	HR	30000	5
11	KK	IT	50000	1
9	II	IT	40000	2
7	GG	IT	40000	3
4	DD	IT	60000	4
3	CC	IT	50000	5
1	AA	IT	40000	6

Step-1 dept: IT & HR

Step-1

```
SELECT * FROM Employee;
```

id	fname	dept	salary
1	AA	IT	40000
2	BB	HR	30000
3	CC	IT	50000
4	DD	IT	60000
5	EE	HR	40000
6	FF	HR	50000
7	GG	IT	40000
8	HH	HR	30000
9	II	IT	40000
10	JJ	HR	30000
11	KK	IT	50000

```
SELECT *,
    ROW_NUMBER()
    OVER(
        PARTITION BY dept
        ORDER BY fname DESC
    ) AS 'ROW_NUMBER'
FROM Employee;
```

Step-2

Step-3

Step-4

id	fname	dept	salary	ROW_NUMBER
10	JJ	HR	30000	1
8	HH	HR	30000	2
6	FF	HR	50000	3
5	EE	HR	40000	4
2	BB	HR	30000	5
11	KK	IT	50000	1
9	II	IT	40000	2
7	GG	IT	40000	3
4	DD	IT	60000	4
3	CC	IT	50000	5
1	AA	IT	40000	6

Step-4

Step-3

Step-2

```sql
SELECT *,

    ROW_NUMBER()
    OVER(
        PARTITION BY dept
        ORDER BY salary DESC
    )AS 'ROW_NUMBER'

FROM Employee;
```

id	fname	dept	salary	ROW_NUMBER
6	FF	HR	50000	1
5	EE	HR	40000	2
10	JJ	HR	30000	3
8	HH	HR	30000	4
2	BB	HR	30000	5
4	DD	IT	60000	1
3	CC	IT	50000	2
11	KK	IT	50000	3
1	AA	IT	40000	4
9	II	IT	40000	5
7	GG	IT	40000	6

Step-1

```sql
SELECT * FROM Employee;
```

id	fname	dept	salary
1	AA	IT	40000
2	BB	HR	30000
3	CC	IT	50000
4	DD	IT	60000
5	EE	HR	40000
6	FF	HR	50000
7	GG	IT	40000
8	HH	HR	30000
9	II	IT	40000
10	JJ	HR	30000
11	KK	IT	50000

```sql
SELECT *,

    ROW_NUMBER()
    OVER(
        PARTITION BY dept
        ORDER BY salary DESC
    )AS 'ROW_NUMBER'

FROM Employee;
```

Step-2

Step-3

Step-4

HR → salary

30000, 40000, 50000
30000, 30000

desc

id	fname	dept	salary	ROW_NUMBER
6	FF	HR	50000	1
5	EE	HR	40000	2
10	JJ	HR	30000	3
8	HH	HR	30000	4
2	BB	HR	30000	5
4	DD	IT	60000	1
3	CC	IT	50000	2
11	KK	IT	50000	3
1	AA	IT	40000	4
9	II	IT	40000	5
7	GG	IT	40000	6

Step-2 Step-3 Step-4

Same IT dept salary desc

```sql
--RANK

SELECT *,
    RANK()
    OVER(
        ORDER BY (SELECT (NULL))
    )AS 'RANK'

FROM Employee;
```

id	fname	dept	salary	RANK
1	AA	IT	40000	1
2	BB	HR	30000	1
3	CC	IT	50000	1
4	DD	IT	60000	1
5	EE	HR	40000	1
6	FF	HR	50000	1
7	GG	IT	40000	1
8	HH	HR	30000	1
9	II	IT	40000	1
10	JJ	HR	30000	1
11	KK	IT	50000	1

```sql
SELECT *,

    RANK()
    OVER(
        ORDER BY dept ASC
    )AS 'RANK'

FROM Employee;
```

id	fname	dept	salary	RANK
2	BB	HR	30000	1
5	EE	HR	40000	1
6	FF	HR	50000	1
8	HH	HR	30000	1
10	JJ	HR	30000	1
11	KK	IT	50000	6
9	II	IT	40000	6
7	GG	IT	40000	6
3	CC	IT	50000	6
4	DD	IT	60000	6
1	AA	IT	40000	6

```sql
SELECT *,

    RANK()
    OVER(
        PARTITION BY dept
        ORDER BY salary DESC
    )AS 'RANK'

FROM Employee;
```

id	fname	dept	salary	RANK
6	FF	HR	50000	1
5	EE	HR	40000	2
10	JJ	HR	30000	3
8	HH	HR	30000	3
2	BB	HR	30000	3
4	DD	IT	60000	1
3	CC	IT	50000	2
11	KK	IT	50000	2
1	AA	IT	40000	4
9	II	IT	40000	4
7	GG	IT	40000	4

```sql
--DENSE_RANK

SELECT *,
    DENSE_RANK()
    OVER(
        ORDER BY (SELECT (NULL))
    )AS 'DENSE_RANK'

FROM Employee;
```

id	fname	dept	salary	DENSE_RANK
1	AA	IT	40000	1
2	BB	HR	30000	1
3	CC	IT	50000	1
4	DD	IT	60000	1
5	EE	HR	40000	1
6	FF	HR	50000	1
7	GG	IT	40000	1
8	HH	HR	30000	1
9	II	IT	40000	1
10	JJ	HR	30000	1
11	KK	IT	50000	1

```sql
SELECT *,

    DENSE_RANK()
    OVER(
        ORDER BY dept ASC
    )AS 'DENSE_RANK'

FROM Employee;
```

id	fname	dept	salary	DENSE_RANK
2	BB	HR	30000	1
5	EE	HR	40000	1
6	FF	HR	50000	1
8	HH	HR	30000	1
10	JJ	HR	30000	1
11	KK	IT	50000	2
9	II	IT	40000	2
7	GG	IT	40000	2
3	CC	IT	50000	2
4	DD	IT	60000	2
1	AA	IT	40000	2

```sql
SELECT *,

    DENSE_RANK()
    OVER(
        PARTITION BY dept
        ORDER BY salary DESC
    )AS 'DENSE_RANK'

FROM Employee;
```

id	fname	dept	salary	DENSE_RANK
6	FF	HR	50000	1
5	EE	HR	40000	2
10	JJ	HR	30000	3
8	HH	HR	30000	3
2	BB	HR	30000	3
4	DD	IT	60000	1
3	CC	IT	50000	2
11	KK	IT	50000	2
1	AA	IT	40000	3
9	II	IT	40000	3
7	GG	IT	40000	3

```sql
SELECT *,
    ROW_NUMBER()
    OVER(
        ORDER BY dept ASC
    )AS 'ROW_NUMBER',

    RANK()
    OVER(
        ORDER BY dept ASC
    )AS 'RANK',

    DENSE_RANK()
    OVER(
        ORDER BY dept ASC
    )AS 'DENSE_RANK'

FROM Employee;
```

id	fname	dept	salary	ROW_NUMBER	RANK	DENSE_RANK
2	BB	HR	30000	1	1	1
5	EE	HR	40000	2	1	1
6	FF	HR	50000	3	1	1
8	HH	HR	30000	4	1	1
10	JJ	HR	30000	5	1	1
11	KK	IT	50000	6	6	2
9	II	IT	40000	7	6	2
7	GG	IT	40000	8	6	2
3	CC	IT	50000	9	6	2
4	DD	IT	60000	10	6	2
1	AA	IT	40000	11	6	2

```sql
SELECT *,
    ROW_NUMBER()
    OVER(
        PARTITION BY dept
        ORDER BY salary DESC
    )AS 'ROW_NUMBER',

    RANK()
    OVER(
        PARTITION BY dept
        ORDER BY salary DESC
    )AS 'RANK',

    DENSE_RANK()
    OVER(
        PARTITION BY dept
        ORDER BY salary DESC
    )AS 'DENSE_RANK'

FROM Employee;
```

id	fname	dept	salary	ROW_NUMBER	RANK	DENSE_RANK
6	FF	HR	50000	1	1	1
5	EE	HR	40000	2	2	2
10	JJ	HR	30000	3	3	3
8	HH	HR	30000	4	3	3
2	BB	HR	30000	5	3	3
4	DD	IT	60000	1	1	1
3	CC	IT	50000	2	2	2
11	KK	IT	50000	3	2	2
1	AA	IT	40000	4	4	3
9	II	IT	40000	5	4	3
7	GG	IT	40000	6	4	3

```
LAG(column, offset) → Previous row value.

LEAD(column, offset) → Next row value.


FIRST_VALUE(column) → First value in partition.

LAST_VALUE(column) → Last value in partition
```

```sql
--LAG

SELECT

     fname
    ,dept
    ,salary

    ,LAG(salary,1)
        OVER(
            PARTITION BY dept
            ORDER BY salary DESC
        ) AS 'Previous ONE row value'

    ,LAG(salary,2)
        OVER(
            PARTITION BY dept
            ORDER BY salary DESC
        ) AS 'Previous TWO row value'
FROM Employee;
```

fname	dept	salary	Previous ONE row value	Previous TWO row value
FF	HR	50000	NULL	NULL
EE	HR	40000	50000	NULL
JJ	HR	30000	40000	50000
HH	HR	30000	30000	40000
BB	HR	30000	30000	30000
DD	IT	60000	NULL	NULL
CC	IT	50000	60000	NULL
KK	IT	50000	50000	60000
AA	IT	40000	50000	50000
II	IT	40000	40000	50000
GG	IT	40000	40000	40000

```sql
--LEAD

SELECT

     fname
    ,dept
    ,salary

    ,LEAD(salary,1)
    OVER(
        PARTITION BY dept
        ORDER BY salary DESC
    ) AS 'Next ONE row value'

    ,LEAD(salary,2)
    OVER(
        PARTITION BY dept
        ORDER BY salary DESC
    ) AS 'Next TWO row value'

FROM Employee;
```

fname	dept	salary	Next ONE row value	Next TWO row value
FF	HR	50000	40000	30000
EE	HR	40000	30000	30000
JJ	HR	30000	30000	30000
HH	HR	30000	30000	NULL
BB	HR	30000	NULL	NULL
DD	IT	60000	50000	50000
CC	IT	50000	50000	40000
KK	IT	50000	40000	40000
AA	IT	40000	40000	40000
II	IT	40000	40000	NULL
GG	IT	40000	NULL	NULL

```sql
--FIRST_VALUE
SELECT

     fname
    ,dept
    ,salary

    ,FIRST_VALUE(salary)
        OVER(
            PARTITION BY dept
            ORDER BY salary DESC
        ) AS 'First value in partition'


FROM Employee;
```

fname	dept	salary	First value in partition
FF	HR	50000	50000
EE	HR	40000	50000
JJ	HR	30000	50000
HH	HR	30000	50000
BB	HR	30000	50000
DD	IT	60000	60000
CC	IT	50000	60000
KK	IT	50000	60000
AA	IT	40000	60000
II	IT	40000	60000
GG	IT	40000	60000

```sql
--LAST_VALUE

SELECT

     fname
    ,dept
    ,salary


    ,LAST_VALUE(salary)
        OVER(
            PARTITION BY dept
            ORDER BY salary DESC
             ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
        ) AS 'Last value in partition'

FROM Employee;
```

fname	dept	salary	Last value in partition
FF	HR	50000	30000
EE	HR	40000	30000
JJ	HR	30000	30000
HH	HR	30000	30000
BB	HR	30000	30000
DD	IT	60000	40000
CC	IT	50000	40000
KK	IT	50000	40000
AA	IT	40000	40000
II	IT	40000	40000
GG	IT	40000	40000