



```

33      -- OVER() CLAUSE
34      select emp_no, department, salary, AVG(salary) OVER() AS company_AVG_salary from employees
35

```



| Result Grid | | | | |
|--|--------|-------------|--------|--------------------|
| Filter Rows: <input type="text"/> | | | | |
| Export:  Wrap Cell Content:  | | | | |
| | emp_no | department | salary | company_AVG_salary |
| ▶ | 1 | engineering | 80000 | 68428.5714 |
| | 2 | engineering | 69000 | 68428.5714 |
| | 3 | engineering | 70000 | 68428.5714 |
| | 4 | engineering | 103000 | 68428.5714 |
| | 5 | engineering | 67000 | 68428.5714 |
| | 6 | engineering | 89000 | 68428.5714 |
| | 7 | engineering | 91000 | 68428.5714 |
| | 8 | sales | 59000 | 68428.5714 |
| | 9 | sales | 70000 | 68428.5714 |
| | 10 | sales | 159000 | 68428.5714 |
| | 11 | sales | 72000 | 68428.5714 |
| | 12 | sales | 60000 | 68428.5714 |

Limit to 1000 rows

```

36      select emp_no, department, salary,
37      MIN(salary) OVER() AS company_minimum_salary,
38      MAX(salary) OVER() AS company_maximum_salary
39      from employees
40
41

```

| Result Grid | | | | | |
|--|--------|-------------|--------|----------------|----------------|
| Filter Rows: <input type="text"/> | | | | | |
| Export:  Wrap Cell Content:  | | | | | |
| | emp_no | department | salary | minimum_salary | maximum_salary |
| ▶ | 1 | engineering | 80000 | 31000 | 159000 |
| | 2 | engineering | 69000 | 31000 | 159000 |
| | 3 | engineering | 70000 | 31000 | 159000 |
| | 4 | engineering | 103000 | 31000 | 159000 |
| | 5 | engineering | 67000 | 31000 | 159000 |
| | 6 | engineering | 89000 | 31000 | 159000 |
| | 7 | engineering | 91000 | 31000 | 159000 |
| | 8 | sales | 59000 | 31000 | 159000 |
| | 9 | sales | 70000 | 31000 | 159000 |
| | 10 | sales | 159000 | 31000 | 159000 |
| | 11 | sales | 72000 | 31000 | 159000 |
| | 12 | sales | 60000 | 31000 | 159000 |

Limit to 1000 rows

```

42 -- Calculate the AVG, MIN and MAX salary for each department, I use OVER(PARTITION BY TO PRODUCE A RESULT FOR EACH ROW OF THE TABLE
43 select emp_no, department, salary,
44        AVG(salary) OVER(PARTITION BY department) AS dept_AVG_salary,
45        AVG(salary) OVER() AS company_salary_AVG,
46        MIN(salary) OVER(PARTITION BY department) AS dept_MIN_salary,
47        MIN(salary) OVER() AS company_MIN_salary,
48        MAX(salary) OVER(PARTITION BY department) AS dept_MAX_salary,
49        MAX(salary) OVER() AS company_MAX_salary
50 FROM employees

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

| emp_no | department | salary | dept_AVG_salary | company_salary_AVG | dept_MIN_salary | company_MIN_salary | dept_MAX_salary | company_MAX_salary |
|--------|------------------|--------|-----------------|--------------------|-----------------|--------------------|-----------------|--------------------|
| 19 | customer service | 31000 | 46571.4286 | 68428.5714 | 31000 | 31000 | 61000 | 159000 |
| 20 | customer service | 56000 | 46571.4286 | 68428.5714 | 31000 | 31000 | 61000 | 159000 |
| 21 | customer service | 55000 | 46571.4286 | 68428.5714 | 31000 | 31000 | 61000 | 159000 |
| 1 | engineering | 80000 | 81285.7143 | 68428.5714 | 67000 | 31000 | 103000 | 159000 |
| 2 | engineering | 69000 | 81285.7143 | 68428.5714 | 67000 | 31000 | 103000 | 159000 |
| 3 | engineering | 70000 | 81285.7143 | 68428.5714 | 67000 | 31000 | 103000 | 159000 |
| 4 | engineering | 103000 | 81285.7143 | 68428.5714 | 67000 | 31000 | 103000 | 159000 |
| 5 | engineering | 67000 | 81285.7143 | 68428.5714 | 67000 | 31000 | 103000 | 159000 |

Result 13

```

51 -- I could have used a simple GROUP BY, and I would have gotten only 3 rows, that is the number of departments in the company
52 select department, ROUND(AVG(salary),2) AS rounded_AVG_salary from employees
53 group by department

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

| department | rounded_AVG_salary |
|------------------|--------------------|
| engineering | 81285.71 |
| sales | 77428.57 |
| customer service | 46571.43 |

Result 14

```

54  -- How many employees are in each department (in this case I notice that there are 7 employees for each dept)
55  select emp_no, department, salary,
56  COUNT(emp_no) OVER(PARTITION BY department) AS total_number_of_employees_in_each_department,
57  COUNT(emp_no) OVER() AS company_total_number_of_employees
58  FROM employees

```

| emp_no | department | salary | total_number_of_employees_in_each_department | company_total_number_of_employees |
|--------|------------------|--------|--|-----------------------------------|
| 17 | customer service | 61000 | 7 | 21 |
| 18 | customer service | 40000 | 7 | 21 |
| 19 | customer service | 31000 | 7 | 21 |
| 20 | customer service | 56000 | 7 | 21 |
| 21 | customer service | 55000 | 7 | 21 |
| 1 | engineering | 80000 | 7 | 21 |
| 2 | engineering | 69000 | 7 | 21 |
| 3 | engineering | 70000 | 7 | 21 |
| 4 | engineering | 103000 | 7 | 21 |
| 5 | engineering | 67000 | 7 | 21 |
| 6 | engineering | 89000 | 7 | 21 |

















```

60  -- Let's sum up the payroll for each department
61  select emp_no, department, salary,
62  SUM(salary) OVER() AS company_total_payroll,
63  SUM(salary) OVER(PARTITION BY department) AS total_payroll_in_each_department
64  FROM employees
65

```

| emp_no | department | salary | company_total_payroll | total_payroll_in_each_department |
|--------|------------------|--------|-----------------------|----------------------------------|
| 20 | customer service | 56000 | 1437000 | 326000 |
| 21 | customer service | 55000 | 1437000 | 326000 |
| 1 | engineering | 80000 | 1437000 | 569000 |
| 2 | engineering | 69000 | 1437000 | 569000 |
| 3 | engineering | 70000 | 1437000 | 569000 |
| 4 | engineering | 103000 | 1437000 | 569000 |
| 5 | engineering | 67000 | 1437000 | 569000 |
| 6 | engineering | 89000 | 1437000 | 569000 |
| 7 | engineering | 91000 | 1437000 | 569000 |
| 8 | sales | 59000 | 1437000 | 542000 |
| 9 | sales | 70000 | 1437000 | 542000 |

Limit to 1000 rows

```

66 -- Use ORDER BY inside the OVER() clause to re-order rows within each window. If ORDER BY is used with an AGGREGATE FUNCTION + OVER(
67 -- PARTITION BY, it gives the ROLLING AVG, ROLLING SUM, ROLLING MIN
68 select emp_no, department, salary,
69 SUM(salary) OVER(PARTITION BY department ORDER BY salary DESC) AS rolling_dept_salary,
70 SUM(salary) OVER(PARTITION BY department) AS total_department_salary
71 FROM employees

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

| emp_no | department | salary | rolling_dept_salary | total_department_salary |
|--------|------------------|--------|---------------------|-------------------------|
| 19 | customer service | 31000 | 326000 | 326000 |
| 4 | engineering | 103000 | 103000 | 569000 |
| 7 | engineering | 91000 | 194000 | 569000 |
| 6 | engineering | 89000 | 283000 | 569000 |
| 1 | engineering | 80000 | 363000 | 569000 |
| 3 | engineering | 70000 | 433000 | 569000 |
| 2 | engineering | 69000 | 502000 | 569000 |
| 5 | engineering | 67000 | 569000 | 569000 |
| 10 | sales | 159000 | 159000 | 542000 |
| 11 | sales | 72000 | 231000 | 542000 |
| 9 | sales | 70000 | 301000 | 542000 |
| 12 | sales | 61000 | 412000 | 542000 |

Limit to 1000 rows

```

73 -- RANKING (is not equal to ROW_NUMBER)
74 -- RANK the salary in the company
75 select emp_no,
76 department,
77 salary,
78 RANK() OVER(ORDER BY salary DESC) AS overall_salary_rank
79 from employees

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

| emp_no | department | salary | overall_salary_rank |
|--------|-------------|--------|---------------------|
| 10 | sales | 159000 | 1 |
| 4 | engineering | 103000 | 2 |
| 7 | engineering | 91000 | 3 |
| 6 | engineering | 89000 | 4 |
| 1 | engineering | 80000 | 5 |
| 11 | sales | 72000 | 6 |
| 3 | engineering | 70000 | 7 |
| 9 | sales | 70000 | 7 |
| 2 | engineering | 69000 | 9 |
| 5 | engineering | 67000 | 10 |

```
80 -- Rank the salary by departments, calculate the department salary rank for each row of the table
81 select emp_no, department, salary,
82 RANK() OVER(ORDER BY salary DESC) AS overall_salary_ranking,
83 RANK() OVER(PARTITION BY department ORDER BY salary DESC) AS dept_salary_ranking
84 from employees
85
86
```

| emp_no | department | salary | overall_salary_ranking | dept_salary_ranking |
|--------|------------------|--------|------------------------|---------------------|
| 17 | customer service | 61000 | 11 | 1 |
| 20 | customer service | 56000 | 16 | 2 |
| 21 | customer service | 55000 | 17 | 3 |
| 16 | customer service | 45000 | 18 | 4 |
| 18 | customer service | 40000 | 19 | 5 |
| 15 | customer service | 38000 | 20 | 6 |
| 19 | customer service | 31000 | 21 | 7 |
| 4 | engineering | 103000 | 2 | 1 |
| 7 | engineering | 91000 | 3 | 2 |
| 6 | engineering | 89000 | 4 | 3 |

```
85
86 -- Rank the salary by each department and return the third highest salary in the engineering dptm
87 WITH CTE AS (select emp_no, department, salary,
88 RANK() OVER(PARTITION BY department ORDER BY salary DESC) AS department_salary_ranking
89 from employees
90 )
91
92 select emp_no, department, salary from CTE
93 where (department_salary_ranking = 3) AND (department = 'Engineering')
```

| emp_no | department | salary |
|--------|-------------|--------|
| 6 | engineering | 89000 |

```

95  -- ROW_NUMBER
96  select emp_no, department, salary,
97  ROW_NUMBER() OVER(PARTITION BY department ORDER BY salary DESC) AS dept_row_number,
98  RANK() OVER(PARTITION BY department ORDER BY salary DESC) AS salary_per_department_ranked,
99  RANK() OVER(ORDER BY salary DESC) AS overall_salary_ranked
100 from employees
101 order by department
102

```

| | emp_no | department | salary | dept_row_number | salary_per_department_ranked | overall_salary_ranked |
|---|--------|------------------|--------|-----------------|------------------------------|-----------------------|
| ▶ | 17 | customer service | 61000 | 1 | 1 | 11 |
| | 20 | customer service | 56000 | 2 | 2 | 16 |
| | 21 | customer service | 55000 | 3 | 3 | 17 |
| | 16 | customer service | 45000 | 4 | 4 | 18 |
| | 18 | customer service | 40000 | 5 | 5 | 19 |
| | 15 | customer service | 38000 | 6 | 6 | 20 |
| | 19 | customer service | 31000 | 7 | 7 | 21 |

USING DENNY'S_DINER SCHEMA I answered the following question by means of the window functions instead of using GROUP BY

| | customer_id | price | overall_spent | total_amount_for_each_customer | total_amount_for_each_customer_percentage |
|--|-------------|-------|---------------|--------------------------------|---|
| | B | 10 | 186 | 74 | 39.7849 |
| | B | 12 | 186 | 74 | 39.7849 |
| | B | 12 | 186 | 74 | 39.7849 |
| | C | 12 | 186 | 36 | 19.3548 |
| | C | 12 | 186 | 36 | 19.3548 |

Limit to 1000 rows

```

12
13  -- How many days has each customer visited the restaurant?
14  select customer_id, order_date,
15  COUNT(order_date) OVER() AS overall_visits,
16  COUNT(order_date) OVER(PARTITION BY customer_id) AS number_of_visits_per_customer
17  from sales
18  -- In this case the answer is not correct as I should have included DISTINCT in the aggregate function, but this
19  -- version of MySQL does not support DISTINCT in a window function
20
21

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [A](#)

| | customer_id | order_date | overall_visits | number_of_visits_per_customer |
|--|-------------|------------|----------------|-------------------------------|
| | B | 2021-01-16 | 15 | 6 |
| | B | 2021-02-01 | 15 | 6 |
| | C | 2021-01-01 | 15 | 3 |
| | C | 2021-01-01 | 15 | 3 |
| | C | 2021-01-07 | 15 | 3 |

Limit to 1000 rows

```

21  -- WHAT WAS THE FIRST ITEM FROM THE MENU PURCHASED BY EACH CUSTOMER?
22  WITH CTE AS (select m.product_name, s.order_date, s.customer_id,
23  ROW_NUMBER() OVER(PARTITION BY s.customer_id ORDER BY s.order_date ASC) AS first_item_row_number,
24  RANK() OVER(PARTITION BY s.customer_id ORDER BY s.order_date ASC) AS ranking
25  from menu as m
26  inner join sales as s on m.product_id = s.product_id
27  )
28  select product_name, order_date, customer_id from CTE
29  where ranking = 1
30

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [A](#)

| | product_name | order_date | customer_id |
|---|--------------|------------|-------------|
| ▶ | sushi | 2021-01-01 | A |
| | curry | 2021-01-01 | A |
| | curry | 2021-01-01 | B |
| | ramen | 2021-01-01 | C |
| | ramen | 2021-01-01 | C |

```

4  -- What was the most purchased item from the menu and how many times it was purchased by all customers?
5  select m.product_name, s.customer_id, s.order_date,
6  COUNT(s.order_date) OVER() AS total_number_of_orders,
7  COUNT(s.order_date) OVER(PARTITION BY m.product_name) AS number_of_orders_per_product_name,
8  (COUNT(s.order_date) OVER(PARTITION BY m.product_name)) / (COUNT(s.order_date) OVER()) * 100
9  AS total_orders_against_products_total_orders_percentage
10 from menu as m
11 inner join sales as s on s.product_id = m.product_id
12 order by COUNT(s.order_date) OVER(PARTITION BY m.product_name) DESC
13

```

| Result Grid | | | | | | |
|-----------------------------------|--------------|-------------|------------|------------------------|-----------------------------------|---|
| Filter Rows: <input type="text"/> | | | | | | |
| Export: Wrap Cell Content: | | | | | | |
| | product_name | customer_id | order_date | total_number_of_orders | number_of_orders_per_product_name | total_orders_against_products_total_orders_percentage |
| ▶ | ramen | A | 2021-01-11 | 15 | 8 | 53.3333 |
| | ramen | A | 2021-01-11 | 15 | 8 | 53.3333 |
| | ramen | B | 2021-01-16 | 15 | 8 | 53.3333 |
| | ramen | B | 2021-02-01 | 15 | 8 | 53.3333 |
| | ramen | C | 2021-01-01 | 15 | 8 | 53.3333 |

```

14  -- To answer the above query I could have used a simple Group by
15  select m.product_name, COUNT(s.order_date) AS number_of_orders from menu as m
16  inner join sales as s on s.product_id = m.product_id
17  GROUP BY m.product_name
18  ORDER BY number_of_orders DESC
19  LIMIT 1
20

```

| Result Grid | | |
|--|--------------|------------------|
| Filter Rows: <input type="text"/> | | |
| Export: Wrap Cell Content: Fetch rows: | | |
| | product_name | number_of_orders |
| ▶ | ramen | 8 |

SQL File 1* x

Limit to 1000 rows

```

21  -- WHICH ITEM WAS THE MOST POPULAR FOR EACH CUSTOMERS?
22  select m.product_name, s.customer_id, COUNT(s.order_date) AS number_of_orders,
23  ROW_NUMBER() OVER(PARTITION BY s.customer_id ORDER BY COUNT(s.order_date) DESC) AS sequence,
24  RANK() OVER(PARTITION BY s.customer_id ORDER BY COUNT(s.order_date) DESC) AS ranking
25  from menu as m
26  inner join sales as s on m.product_id = s.product_id
27  group by s.customer_id, m.product_name
28

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [FA](#)

| | product_name | customer_id | number_of_orders | sequence | ranking |
|---|--------------|-------------|------------------|----------|---------|
| ▶ | ramen | A | 3 | 1 | 1 |
| | curry | A | 2 | 2 | 2 |
| | sushi | A | 1 | 3 | 3 |
| | curry | B | 2 | 1 | 1 |
| | sushi | B | 2 | 2 | 1 |
| | ramen | B | 2 | 3 | 1 |
| | ramen | C | 3 | 1 | 1 |

Limit to 1000 rows

```

4  -- WHICH ITEM WAS PURCHASED FIRST BY THE CUSTOMER AFTER THEY BECAME A MEMBER?
5  WITH CTE AS (select mem.customer_id, s.order_date, m.product_name, mem.join_date,
6  ROW_NUMBER() OVER(PARTITION BY mem.customer_id ORDER BY order_date ASC) AS sequence,
7  RANK() OVER(PARTITION BY mem.customer_id ORDER BY order_date ASC) AS ranking
8  from menu as m
9  inner join sales as s on m.product_id = s.product_id
10 inner join members as mem on mem.customer_id = s.customer_id
11 where s.order_date >= mem.join_date)
12 select product_name, customer_id from CTE
13 where sequence = 1

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [FA](#)

| | product_name | customer_id |
|---|--------------|-------------|
| ▶ | curry | A |
| | sushi | B |

Limit to 1000 rows

```

14
15  -- What is the total items and amount spent for each member before they became a member?
16  select s.order_date, mem.customer_id, mem.join_date, s.product_id, m.price,
17         COUNT(s.product_id) OVER(PARTITION BY mem.customer_id) AS total_items_for_each_member,
18         SUM(m.price) OVER(PARTITION BY mem.customer_id) AS total_amount_for_each_member
19  from menu as m
20  inner join sales as s on m.product_id = s.product_id
21  inner join members as mem on mem.customer_id = s.customer_id
22  where s.order_date < mem.join_date
23

```

Result Grid

Filter Rows:

Export:

Wrap Cell Content: ☐

| | order_date | customer_id | join_date | product_id | price | total_items_for_each_member | total_amount_for_each_member |
|---|------------|-------------|------------|------------|-------|-----------------------------|------------------------------|
| ▶ | 2021-01-01 | A | 2021-01-07 | 1 | 10 | 2 | 25 |
| | 2021-01-01 | A | 2021-01-07 | 2 | 15 | 2 | 25 |
| | 2021-01-01 | B | 2021-01-09 | 2 | 15 | 3 | 40 |
| | 2021-01-02 | B | 2021-01-09 | 2 | 15 | 3 | 40 |
| | 2021-01-04 | B | 2021-01-09 | 1 | 10 | 3 | 40 |

Result 18 ×