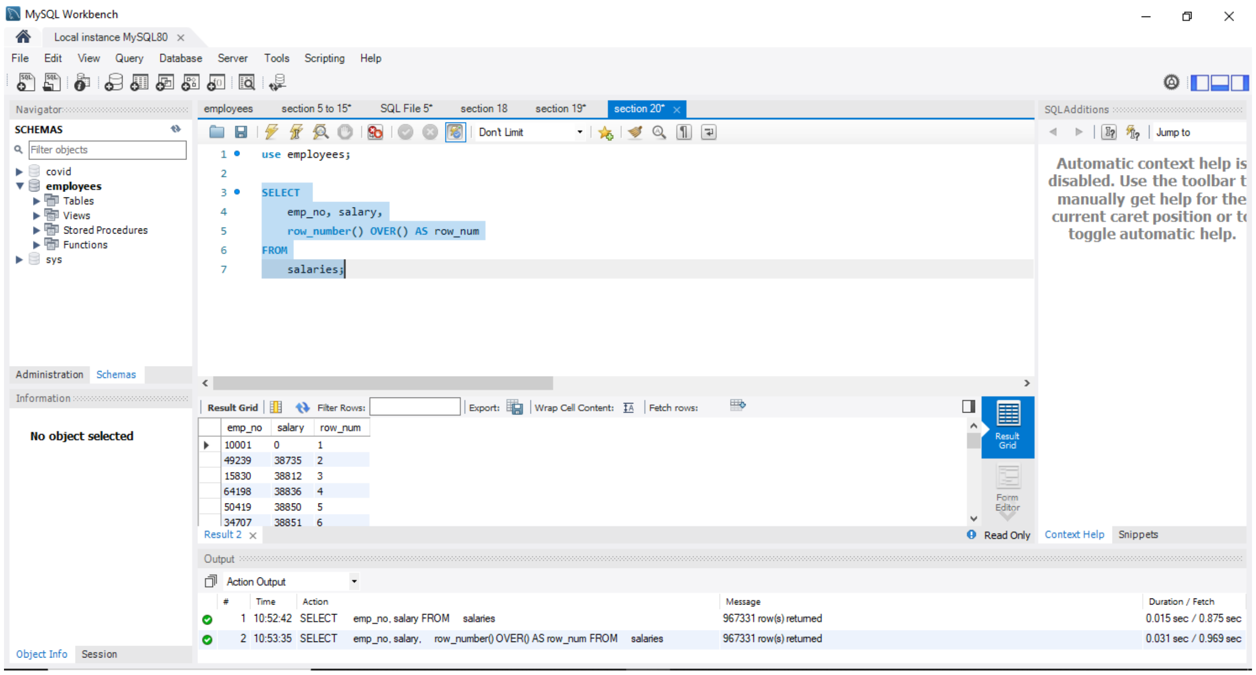
SECTION 20

We discussed MySQL Windows Functions in this section.

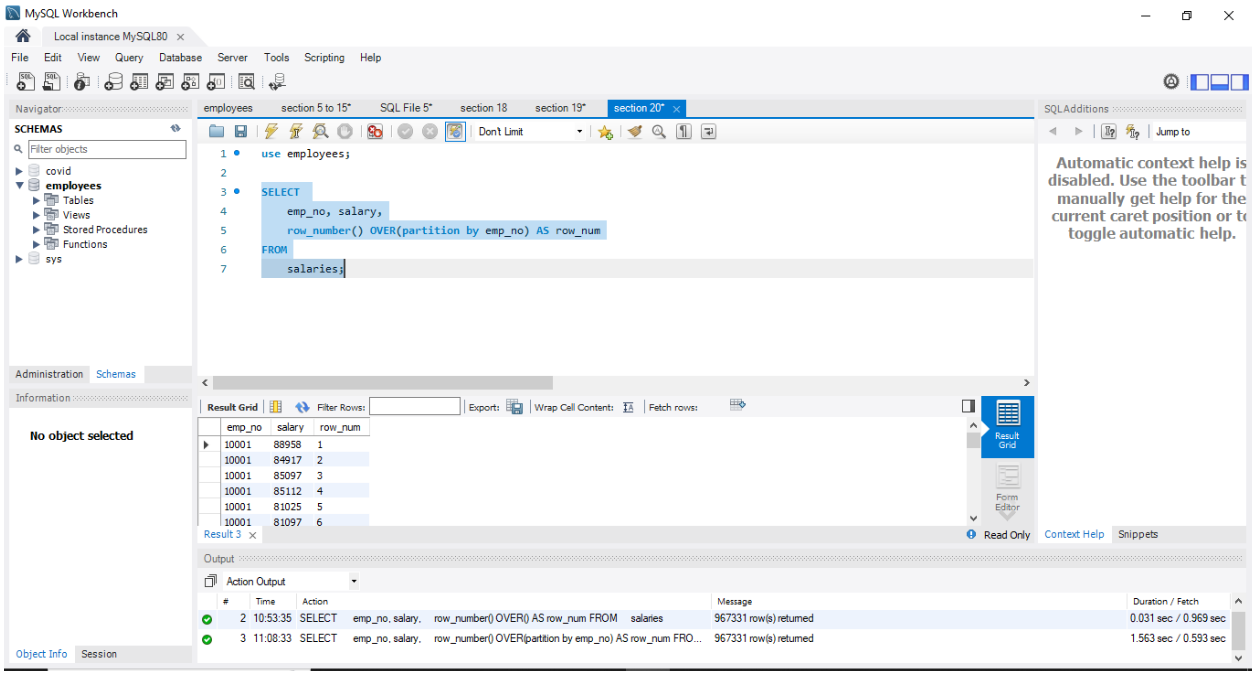


```sql
use employees;

SELECT
    emp_no, salary,
    row_number() OVER() AS row_num
FROM
    salaries;
```

emp_no	salary	row_num
10001	0	1
49239	38735	2
15830	38812	3
64198	38836	4
50419	38850	5
34707	38851	6

```
#  Time      Action                                                        Message                    Duration / Fetch
1  10:52:42  SELECT  emp_no, salary FROM  salaries                         967331 row(s) returned     0.015 sec / 0.875 sec
2  10:53:35  SELECT  emp_no, salary,  row_number() OVER() AS row_num FROM  salaries   967331 row(s) returned   0.031 sec / 0.969 sec
```



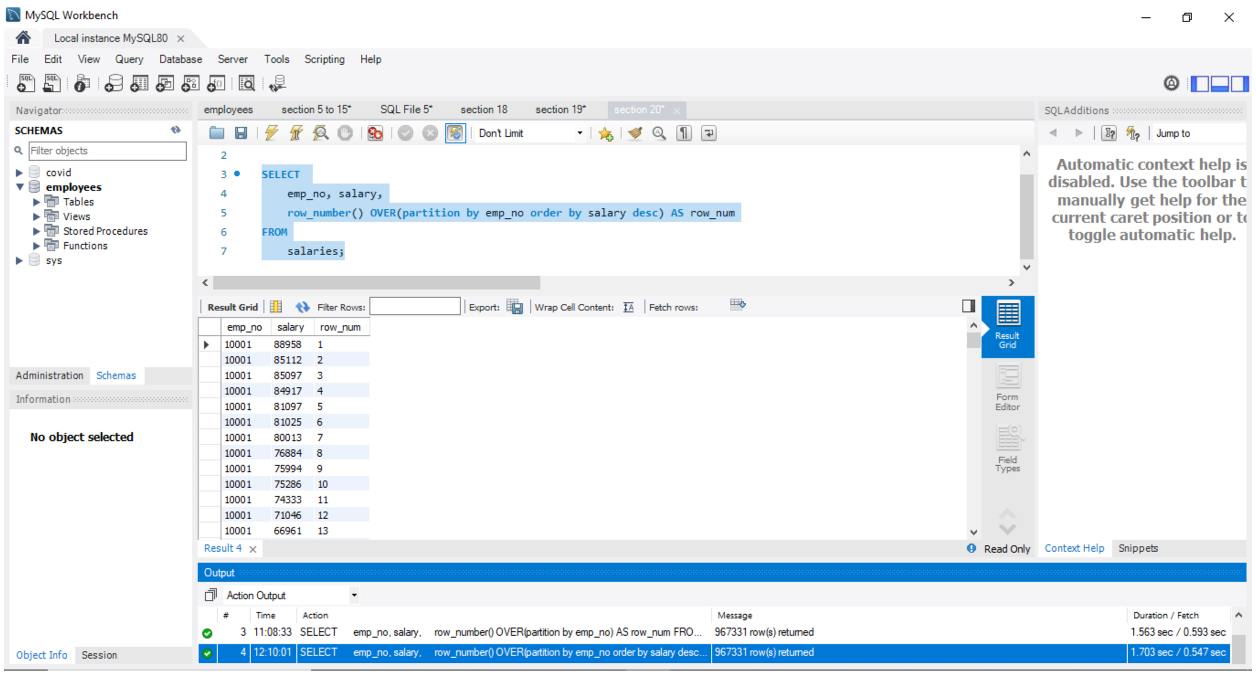
```sql
use employees;

SELECT
    emp_no, salary,
    row_number() OVER(partition by emp_no) AS row_num
FROM
    salaries;
```

emp_no	salary	row_num
10001	88958	1
10001	84917	2
10001	85097	3
10001	85112	4
10001	81025	5
10001	81097	6

```
#  Time      Action                                                                          Message                  Duration / Fetch
2  10:53:35  SELECT  emp_no, salary,  row_number() OVER() AS row_num FROM  salaries          967331 row(s) returned   0.031 sec / 0.969 sec
3  11:08:33  SELECT  emp_no, salary,  row_number() OVER(partition by emp_no) AS row_num FRO...   967331 row(s) returned   1.563 sec / 0.593 sec
```

Now, if we want to rank our salaries in descending order:



```sql
SELECT
    emp_no, salary,
    row_number() OVER(partition by emp_no order by salary desc) AS row_num
FROM
    salaries;
```

emp_no	salary	row_num
10001	88958	1
10001	85112	2
10001	85097	3
10001	84917	4
10001	81097	5
10001	81025	6
10001	80013	7
10001	76884	8
10001	75994	9
10001	75286	10
10001	74333	11
10001	71046	12
10001	66961	13

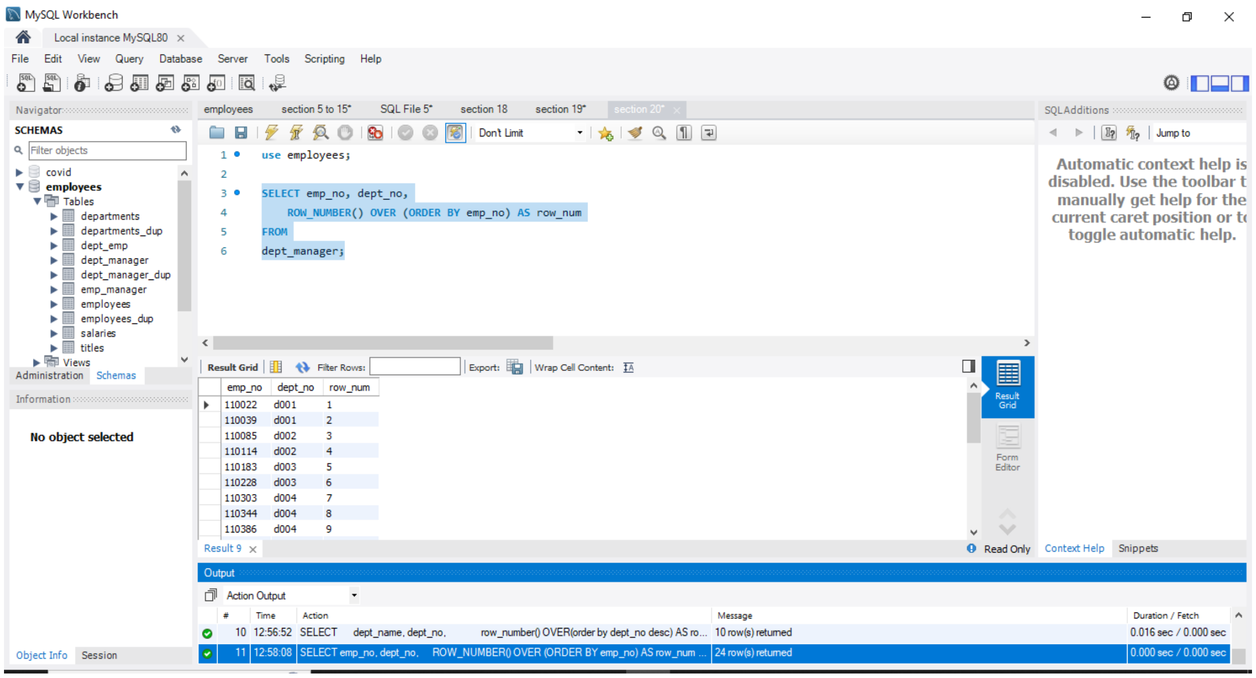
```
#  Time      Action                                                                        Message                  Duration / Fetch
3  11:08:33  SELECT  emp_no, salary,  row_number() OVER(partition by emp_no) AS row_num FRO...  967331 row(s) returned  1.563 sec / 0.593 sec
4  12:10:01  SELECT  emp_no, salary,  row_number() OVER(partition by emp_no order by salary desc...  967331 row(s) returned  1.703 sec / 0.547 sec
```

Exercise #1:

Write a query that upon execution, assigns a row number to all managers we have information for in the "employees" database (regardless of their department).

Let the numbering disregard the department the managers have worked in. Also, let it start from the value of 1. Assign that value to the manager with the lowest employee number.

Solution:



```
use employees;

SELECT emp_no, dept_no,
    ROW_NUMBER() OVER (ORDER BY emp_no) AS row_num
FROM
dept_manager;
```

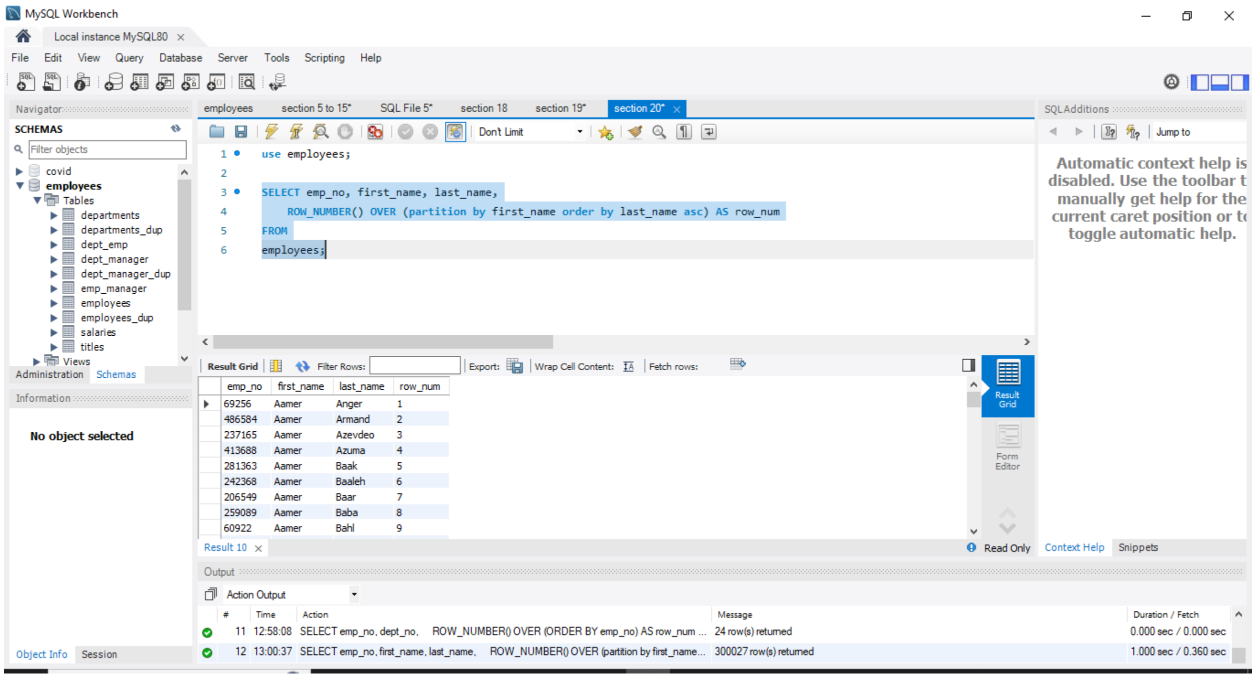
emp_no	dept_no	row_num
110022	d001	1
110039	d001	2
110085	d002	3
110114	d002	4
110183	d003	5
110228	d003	6
110303	d004	7
110344	d004	8
110386	d004	9

```
#   Time      Action                                                                   Message            Duration / Fetch
10  12:56:52  SELECT    dept_name, dept_no,     row_number() OVER(order by dept_no desc) AS ro...  10 row(s) returned  0.016 sec / 0.000 sec
11  12:58:08  SELECT emp_no, dept_no,   ROW_NUMBER() OVER (ORDER BY emp_no) AS row_num ...   24 row(s) returned  0.000 sec / 0.000 sec
```

Exercise #2:

Write a query that upon execution, assigns a sequential number for each employee number registered in the "employees" table. Partition the data by the employee's first name and order it by their last name in ascending order (for each partition).

Solution:



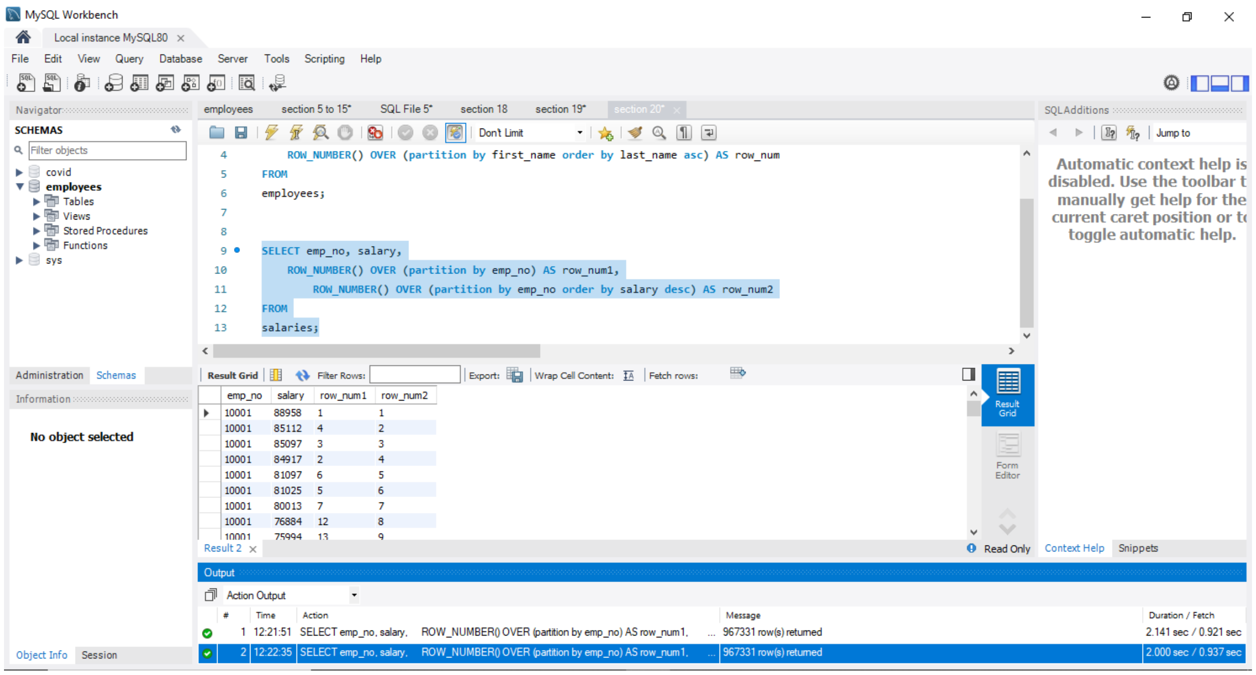
```
use employees;

SELECT emp_no, first_name, last_name,
    ROW_NUMBER() OVER (partition by first_name order by last_name asc) AS row_num
FROM
employees;
```

emp_no	first_name	last_name	row_num
69256	Aamer	Anger	1
486584	Aamer	Armand	2
237165	Aamer	Azevdeo	3
413688	Aamer	Azuma	4
281363	Aamer	Baak	5
242368	Aamer	Baaleh	6
206549	Aamer	Baar	7
259089	Aamer	Baba	8
60922	Aamer	Bahl	9

```
#   Time      Action                                                                        Message                Duration / Fetch
11  12:58:08  SELECT emp_no, dept_no,   ROW_NUMBER() OVER (ORDER BY emp_no) AS row_num ...       24 row(s) returned      0.000 sec / 0.000 sec
12  13:00:37  SELECT emp_no, first_name, last_name,   ROW_NUMBER() OVER (partition by first_name...  300027 row(s) returned  1.000 sec / 0.360 sec
```

We can also use multiple window functions in a single query. Here is an example:



```
    ROW_NUMBER() OVER (partition by first_name order by last_name asc) AS row_num
FROM
employees;


SELECT emp_no, salary,
    ROW_NUMBER() OVER (partition by emp_no) AS row_num1,
        ROW_NUMBER() OVER (partition by emp_no order by salary desc) AS row_num2
FROM
salaries;
```

```
emp_no  salary  row_num1  row_num2
10001   88958   1         1
10001   85112   4         2
10001   85097   3         3
10001   84917   2         4
10001   81097   6         5
10001   81025   5         6
10001   80013   7         7
10001   76884   12        8
10001   75994   13        9
```

```
#  Time      Action                                                                      Message                  Duration / Fetch
1  12:21:51  SELECT emp_no, salary,   ROW_NUMBER() OVER (partition by emp_no) AS row_num1,  ...  967331 row(s) returned  2.141 sec / 0.921 sec
2  12:22:35  SELECT emp_no, salary,   ROW_NUMBER() OVER (partition by emp_no) AS row_num1,  ...  967331 row(s) returned  2.000 sec / 0.937 sec
```

Exercise #1:

Obtain a result set containing the salary values each manager has signed a contract for. To obtain the data, refer to the "employees" database.

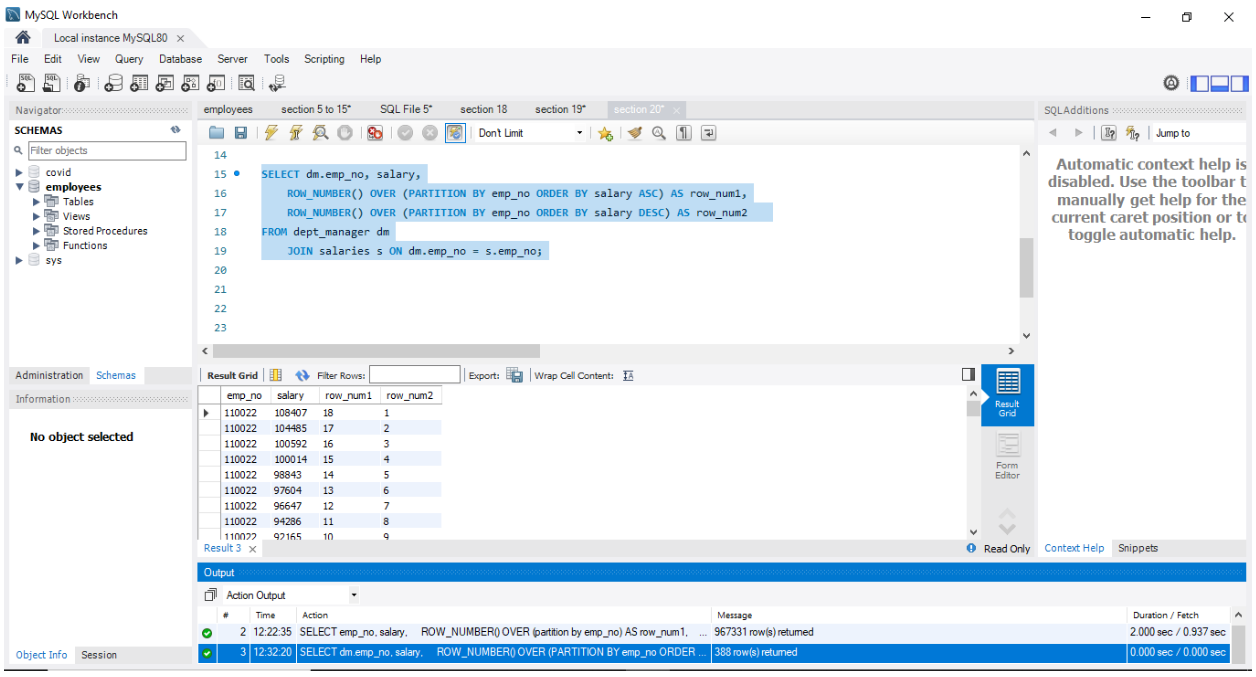
Use window functions to add the following two columns to the final output:

- A column containing the row number of each row from the obtained dataset, starting from 1.

- A column containing the sequential row numbers associated to the rows for each manager, where their highest salary has been given a number equal to the number of rows in the given partition, and their lowest - the number 1.

Finally, while presenting the output, make sure that the data has been ordered by the values in the first of the row number columns, and then by the salary values for each partition in ascending order.

Solution:



```sql
SELECT dm.emp_no, salary,
    ROW_NUMBER() OVER (PARTITION BY emp_no ORDER BY salary ASC) AS row_num1,
    ROW_NUMBER() OVER (PARTITION BY emp_no ORDER BY salary DESC) AS row_num2
FROM dept_manager dm
    JOIN salaries s ON dm.emp_no = s.emp_no;
```

emp_no	salary	row_num1	row_num2
110022	108407	18	1
110022	104485	17	2
110022	100592	16	3
110022	100014	15	4
110022	98843	14	5
110022	97604	13	6
110022	96647	12	7
110022	94286	11	8
110022	92165	10	9

```
#  Time      Action                                                                      Message                   Duration / Fetch
2  12:22:35  SELECT emp_no, salary,   ROW_NUMBER() OVER (partition by emp_no) AS row_num1,  ...  967331 row(s) returned   2.000 sec / 0.937 sec
3  12:32:20  SELECT dm.emp_no, salary,   ROW_NUMBER() OVER (PARTITION BY emp_no ORDER ...   388 row(s) returned       0.000 sec / 0.000 sec
```

Exercise #2:

Obtain a result set containing the salary values each manager has signed a contract for. To obtain the data, refer to the "employees" database.

Use window functions to add the following two columns to the final output:

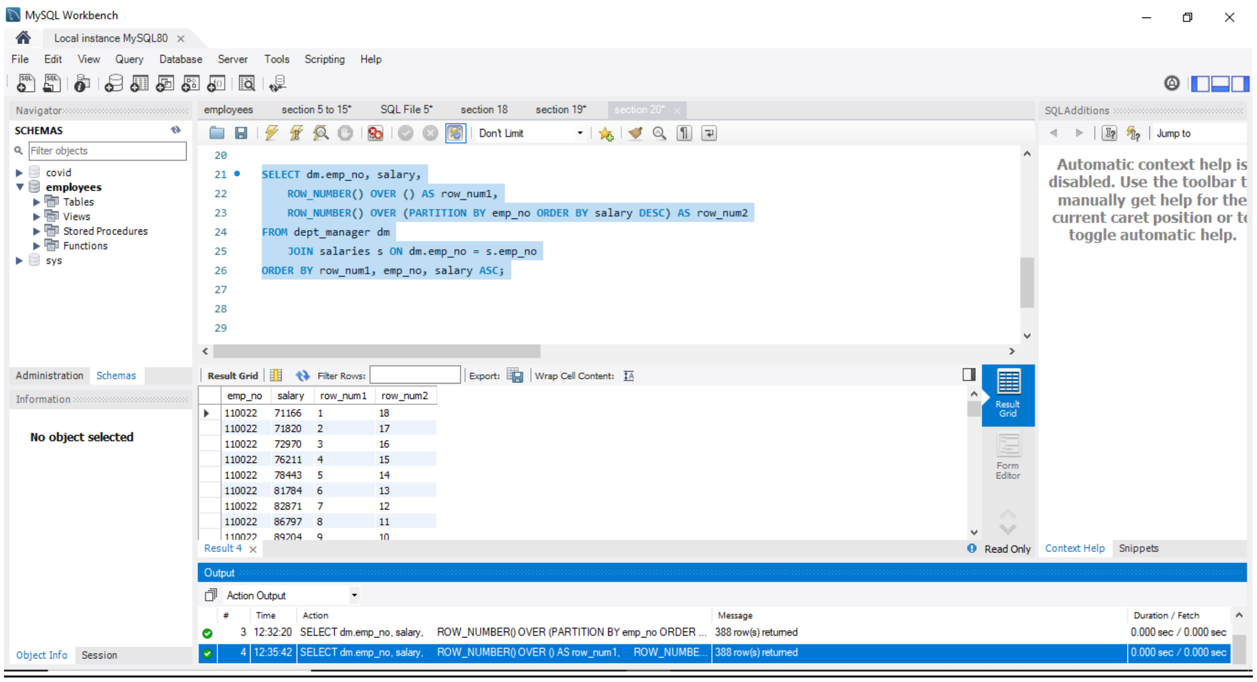
- a column containing the row numbers associated to each manager, where their highest salary has been given a number equal to the number of rows in the given partition, and their lowest - the number 1.

- a column containing the row numbers associated to each manager, where their highest salary has been given the number of 1, and the lowest - a value equal to the number of rows in the given partition.

Let your output be ordered by the salary values associated to each manager in descending order.

Hint: Please note that you don't need to use an ORDER BY clause in your SELECT statement to retrieve the desired output.

Solution:



```sql
SELECT dm.emp_no, salary,
    ROW_NUMBER() OVER () AS row_num1,
    ROW_NUMBER() OVER (PARTITION BY emp_no ORDER BY salary DESC) AS row_num2
FROM dept_manager dm
    JOIN salaries s ON dm.emp_no = s.emp_no
ORDER BY row_num1, emp_no, salary ASC;
```

emp_no	salary	row_num1	row_num2
110022	71166	1	18
110022	71820	2	17
110022	72970	3	16
110022	76211	4	15
110022	78443	5	14
110022	81784	6	13
110022	82871	7	12
110022	86797	8	11
110022	89204	9	10

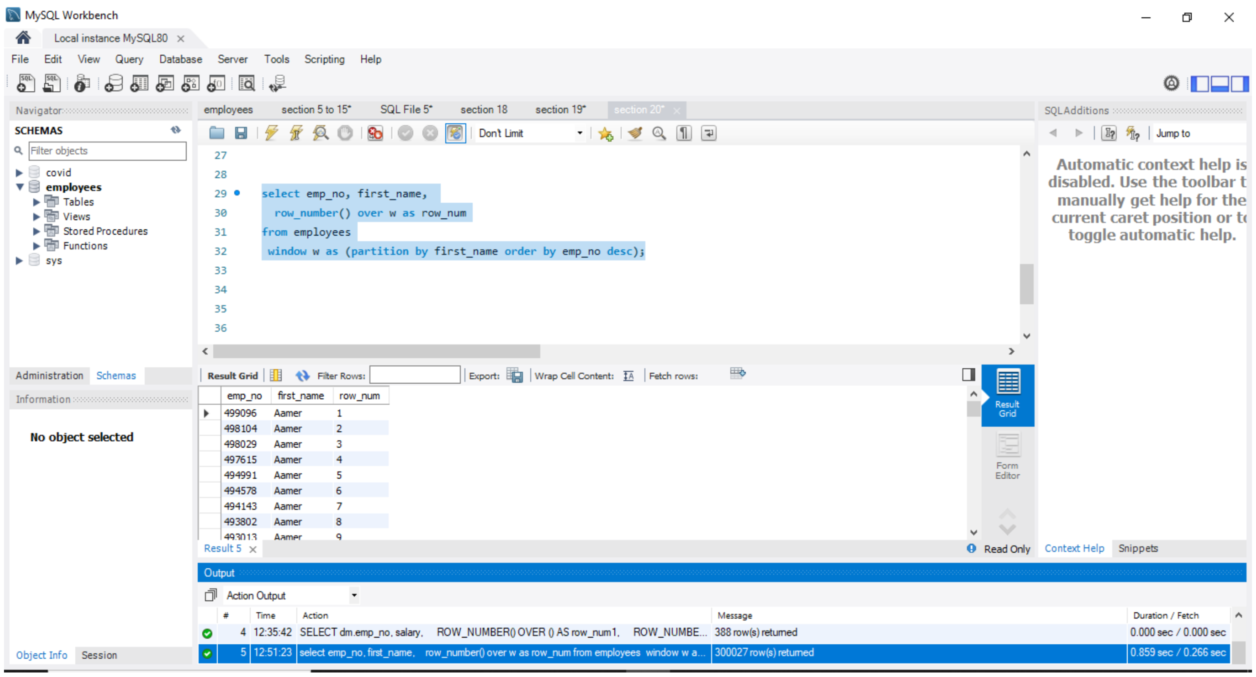
```
#  Time      Action                                                                      Message               Duration / Fetch
3  12:32:20  SELECT dm.emp_no, salary,   ROW_NUMBER() OVER (PARTITION BY emp_no ORDER ...  388 row(s) returned   0.000 sec / 0.000 sec
4  12:35:42  SELECT dm.emp_no, salary,   ROW_NUMBER() OVER () AS row_num1,   ROW_NUMBE...  388 row(s) returned   0.000 sec / 0.000 sec
```

Exercise #1:

Write a query that provides row numbers for all workers from the "employees" table, partitioning the data by their first names and ordering each partition by their employee number in ascending order.

*NB! While writing the desired query, do *not* use an ORDER BY clause in the relevant SELECT statement. At the same time, do use a WINDOW clause to provide the required window specification.*

Solution:



```sql
select emp_no, first_name,
  row_number() over w as row_num
from employees
 window w as (partition by first_name order by emp_no desc);
```

emp_no	first_name	row_num
499096	Aamer	1
498104	Aamer	2
498029	Aamer	3
497615	Aamer	4
494991	Aamer	5
494578	Aamer	6
494143	Aamer	7
493802	Aamer	8
493013	Aamer	9

```
#  Time      Action                                                                          Message                  Duration / Fetch
4  12:35:42  SELECT dm.emp_no, salary,    ROW_NUMBER() OVER () AS row_num1,    ROW_NUMBE...   388 row(s) returned      0.000 sec / 0.000 sec
5  12:51:23  select emp_no, first_name,    row_number() over w as row_num from employees  window w a...   300027 row(s) returned   0.859 sec / 0.266 sec
```

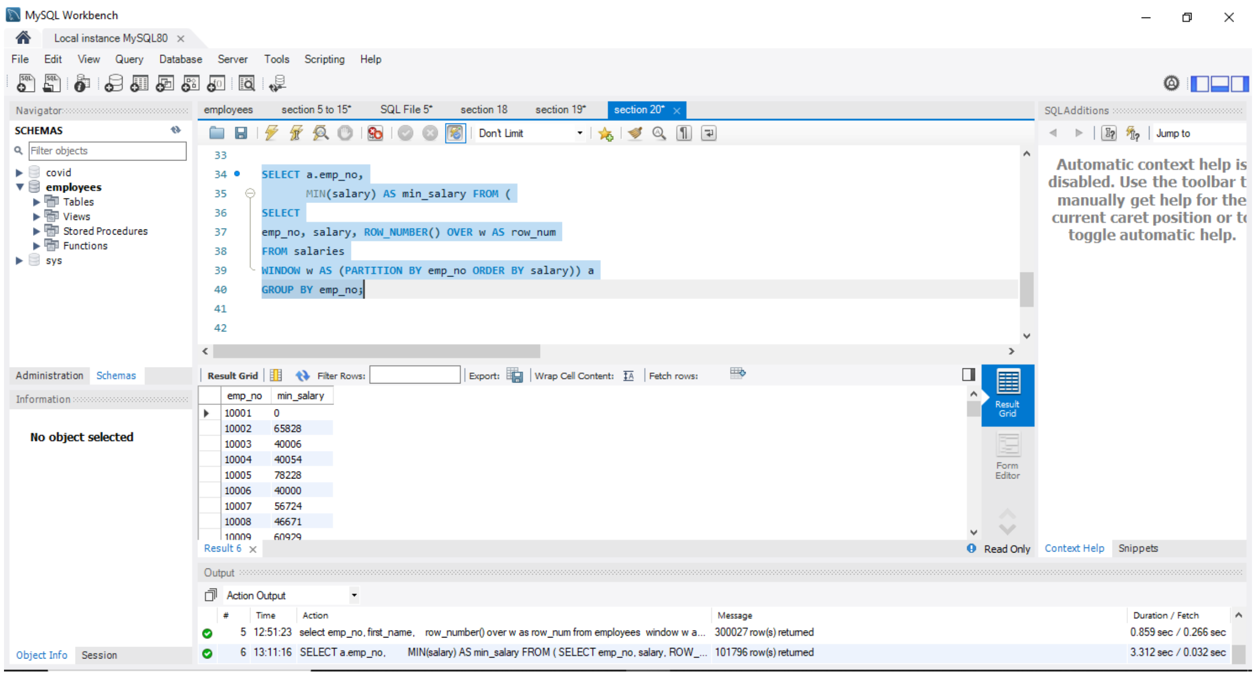
We explored differences between PARTITION BY and GROUP BY clause.

Exercise #1:

Find out the lowest salary value each employee has ever signed a contract for. To obtain the desired output, use a subquery containing a window function, as well as a window specification introduced with the help of the WINDOW keyword.

Also, to obtain the desired result set, refer only to data from the "salaries" table.

<u>Solution:</u>



```sql
SELECT a.emp_no,
       MIN(salary) AS min_salary FROM (
SELECT
emp_no, salary, ROW_NUMBER() OVER w AS row_num
FROM salaries
WINDOW w AS (PARTITION BY emp_no ORDER BY salary)) a
GROUP BY emp_no;
```

emp_no	min_salary
10001	0
10002	65828
10003	40006
10004	40054
10005	78228
10006	40000
10007	56724
10008	46671
10009	60929

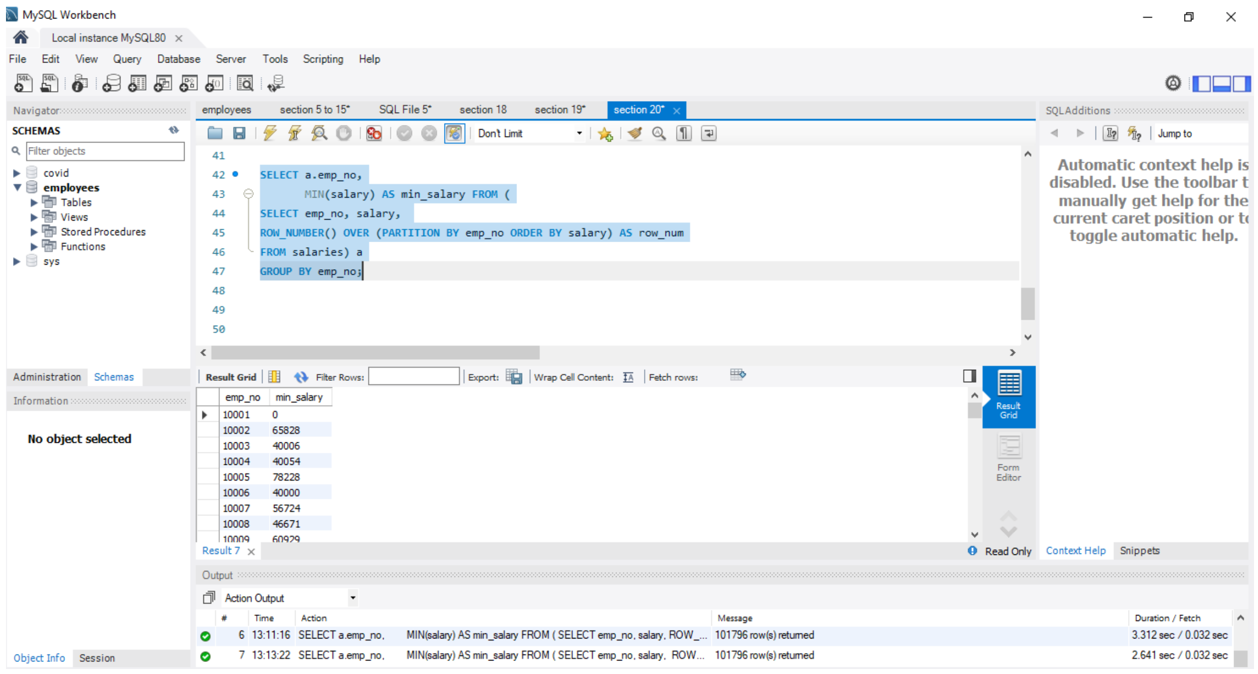
#	Time	Action	Message	Duration / Fetch
5	12:51:23	select emp_no, first_name, row_number() over w as row_num from employees window w a...	300027 row(s) returned	0.859 sec / 0.266 sec
6	13:11:16	SELECT a.emp_no, MIN(salary) AS min_salary FROM (SELECT emp_no, salary, ROW_...	101796 row(s) returned	3.312 sec / 0.032 sec

Exercise #2:

Again, find out the lowest salary value each employee has ever signed a contract for. Once again, to obtain the desired output, use a subquery containing a window function. This time, however, introduce the window specification in the field list of the given subquery.

To obtain the desired result set, refer only to data from the "salaries" table.

Solution:



```sql
SELECT a.emp_no,
       MIN(salary) AS min_salary FROM (
SELECT emp_no, salary,
ROW_NUMBER() OVER (PARTITION BY emp_no ORDER BY salary) AS row_num
FROM salaries) a
GROUP BY emp_no;
```

emp_no	min_salary
10001	0
10002	65828
10003	40006
10004	40054
10005	78228
10006	40000
10007	56724
10008	46671
10009	60929

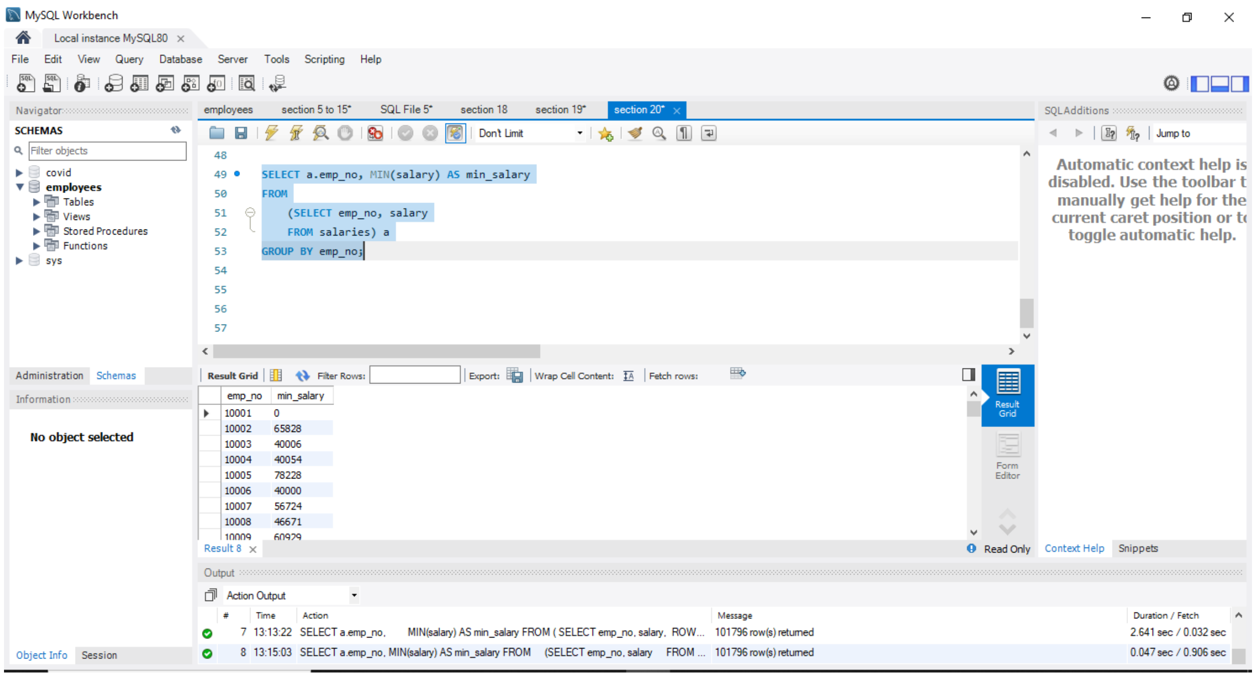
#	Time	Action	Message	Duration / Fetch
6	13:11:16	SELECT a.emp_no, MIN(salary) AS min_salary FROM (SELECT emp_no, salary, ROW_...	101796 row(s) returned	3.312 sec / 0.032 sec
7	13:13:22	SELECT a.emp_no, MIN(salary) AS min_salary FROM (SELECT emp_no, salary, ROW...	101796 row(s) returned	2.641 sec / 0.032 sec

Exercise #3:

Once again, find out the lowest salary value each employee has ever signed a contract for. This time, to obtain the desired output, avoid using a window function. Just use an aggregate function and a subquery.

To obtain the desired result set, refer only to data from the "salaries" table.

Solution:



```sql
SELECT a.emp_no, MIN(salary) AS min_salary
FROM
    (SELECT emp_no, salary
    FROM salaries) a
GROUP BY emp_no;
```

emp_no	min_salary
10001	0
10002	65828
10003	40006
10004	40054
10005	78228
10006	40000
10007	56724
10008	46671
10009	60929

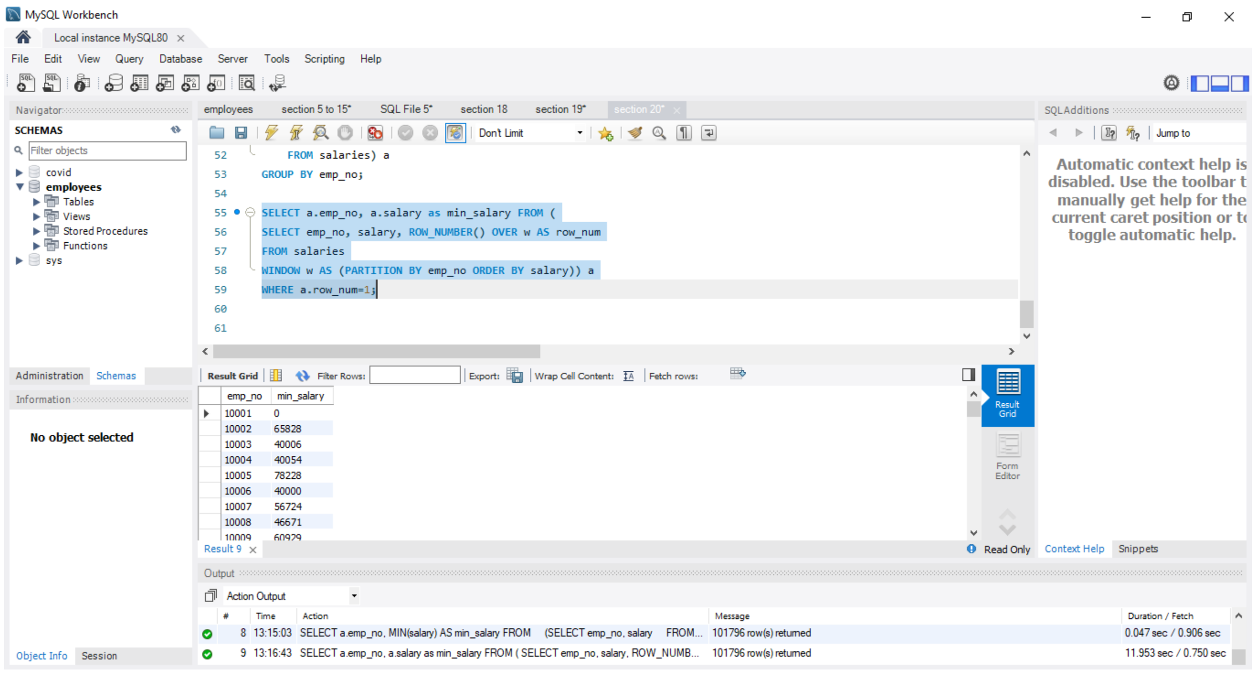
```
#  Time      Action                                                                          Message                  Duration / Fetch
7  13:13:22  SELECT a.emp_no,    MIN(salary) AS min_salary FROM ( SELECT emp_no, salary,  ROW...  101796 row(s) returned   2.641 sec / 0.032 sec
8  13:15:03  SELECT a.emp_no, MIN(salary) AS min_salary FROM    (SELECT emp_no, salary    FROM ...  101796 row(s) returned   0.047 sec / 0.906 sec
```

Exercise #4:

Once more, find out the lowest salary value each employee has ever signed a contract for. To obtain the desired output, use a subquery containing a window function, as well as a window specification introduced with the help of the WINDOW keyword. Moreover, obtain the output without using a GROUP BY clause in the outer query.

To obtain the desired result set, refer only to data from the "salaries" table.

Solution:



```
    FROM salaries) a
GROUP BY emp_no;

SELECT a.emp_no, a.salary as min_salary FROM (
SELECT emp_no, salary, ROW_NUMBER() OVER w AS row_num
FROM salaries
WINDOW w AS (PARTITION BY emp_no ORDER BY salary)) a
WHERE a.row_num=1;
```

emp_no	min_salary
10001	0
10002	65828
10003	40006
10004	40054
10005	78228
10006	40000
10007	56724
10008	46671
10009	60929

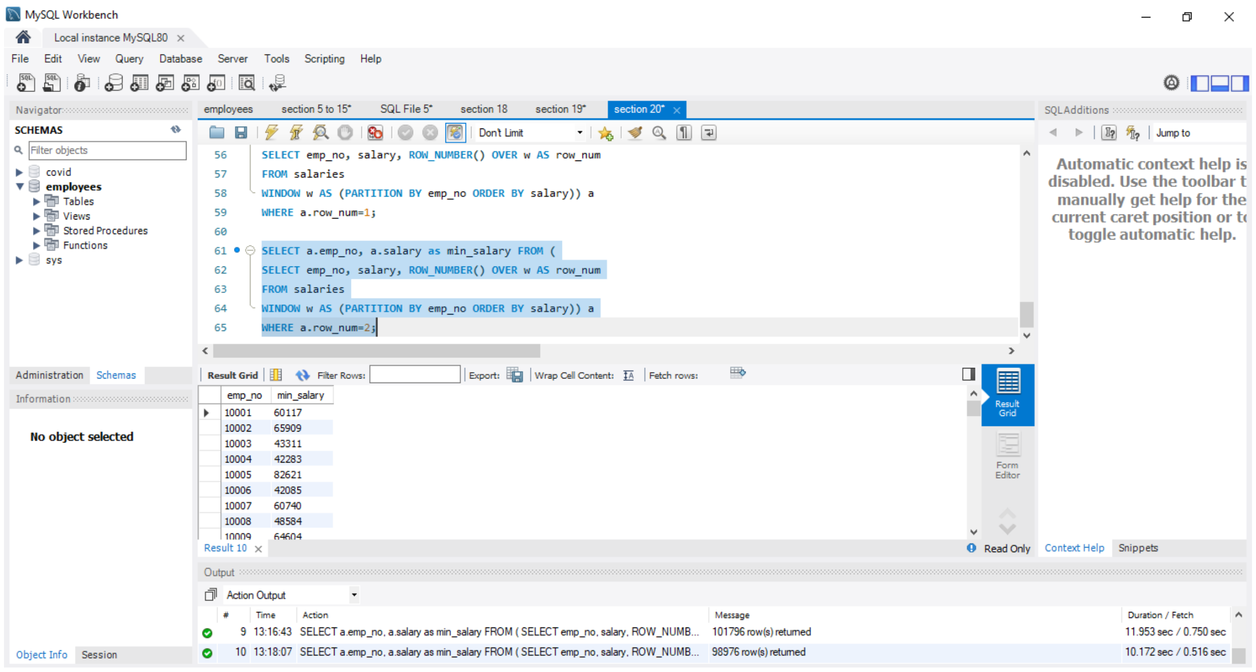
```
#  Time      Action                                                                   Message                   Duration / Fetch
8  13:15:03  SELECT a.emp_no, MIN(salary) AS min_salary FROM   (SELECT emp_no, salary   FROM...  101796 row(s) returned  0.047 sec / 0.906 sec
9  13:16:43  SELECT a.emp_no, a.salary as min_salary FROM ( SELECT emp_no, salary, ROW_NUMB...  101796 row(s) returned  11.953 sec / 0.750 sec
```

Exercise #5:

Find out the second-lowest salary value each employee has ever signed a contract for. To obtain the desired output, use a subquery containing a window function, as well as a window specification introduced with the help of the WINDOW keyword. Moreover, obtain the desired result set without using a GROUP BY clause in the outer query.

To obtain the desired result set, refer only to data from the "salaries" table.

<u>Solution:</u>



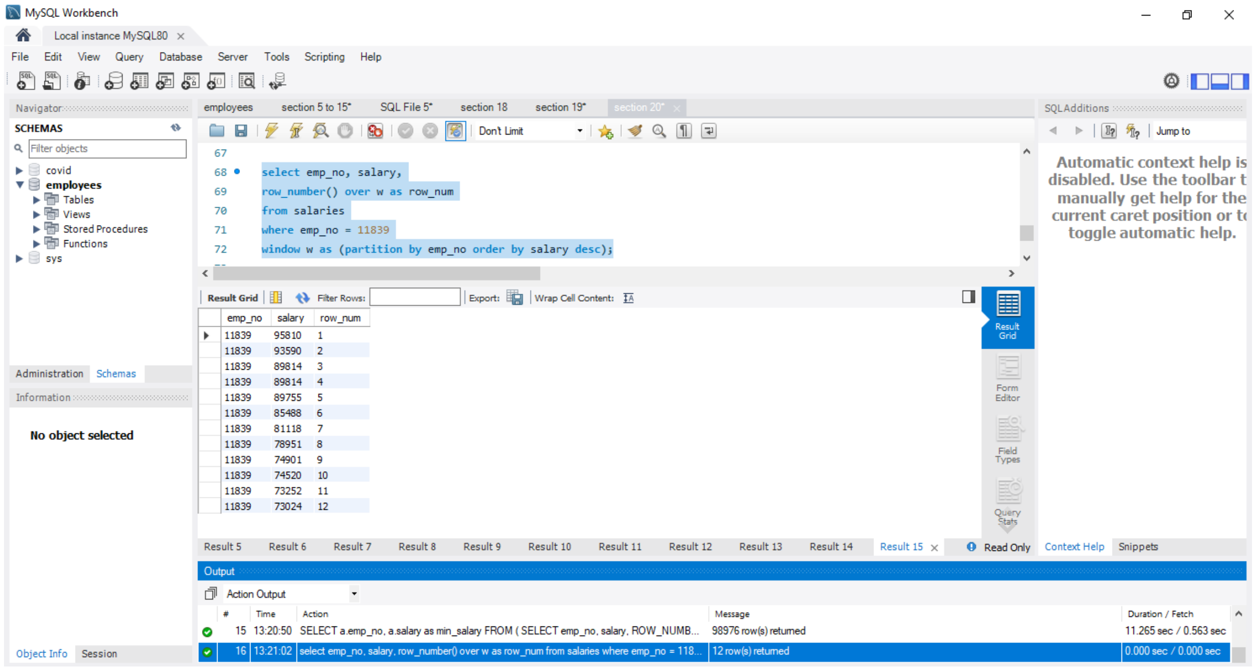
```
SELECT emp_no, salary, ROW_NUMBER() OVER w AS row_num
FROM salaries
WINDOW w AS (PARTITION BY emp_no ORDER BY salary)) a
WHERE a.row_num=1;

SELECT a.emp_no, a.salary as min_salary FROM (
SELECT emp_no, salary, ROW_NUMBER() OVER w AS row_num
FROM salaries
WINDOW w AS (PARTITION BY emp_no ORDER BY salary)) a
WHERE a.row_num=2;
```

emp_no	min_salary
10001	60117
10002	65909
10003	43311
10004	42283
10005	82621
10006	42085
10007	60740
10008	48584
10009	64604

```
#   Time      Action                                                                   Message                  Duration / Fetch
9   13:16:43  SELECT a.emp_no, a.salary as min_salary FROM ( SELECT emp_no, salary, ROW_NUMB...  101796 row(s) returned  11.953 sec / 0.750 sec
10  13:18:07  SELECT a.emp_no, a.salary as min_salary FROM ( SELECT emp_no, salary, ROW_NUMB...  98976 row(s) returned   10.172 sec / 0.516 sec
```

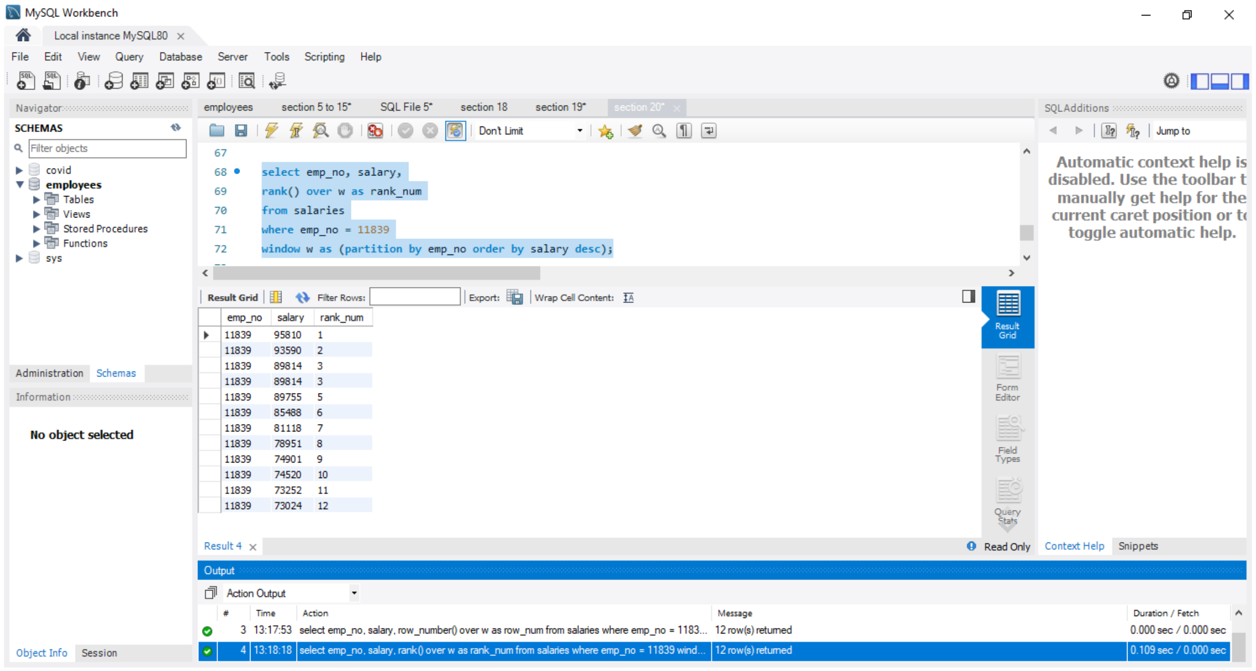
Let’s explore other window functions: RANK() and DENSE_RANK():



```
select emp_no, salary,
row_number() over w as row_num
from salaries
where emp_no = 11839
window w as (partition by emp_no order by salary desc);
```

emp_no	salary	row_num
11839	95810	1
11839	93590	2
11839	89814	3
11839	89814	4
11839	89755	5
11839	85488	6
11839	81118	7
11839	78951	8
11839	74901	9
11839	74520	10
11839	73252	11
11839	73024	12

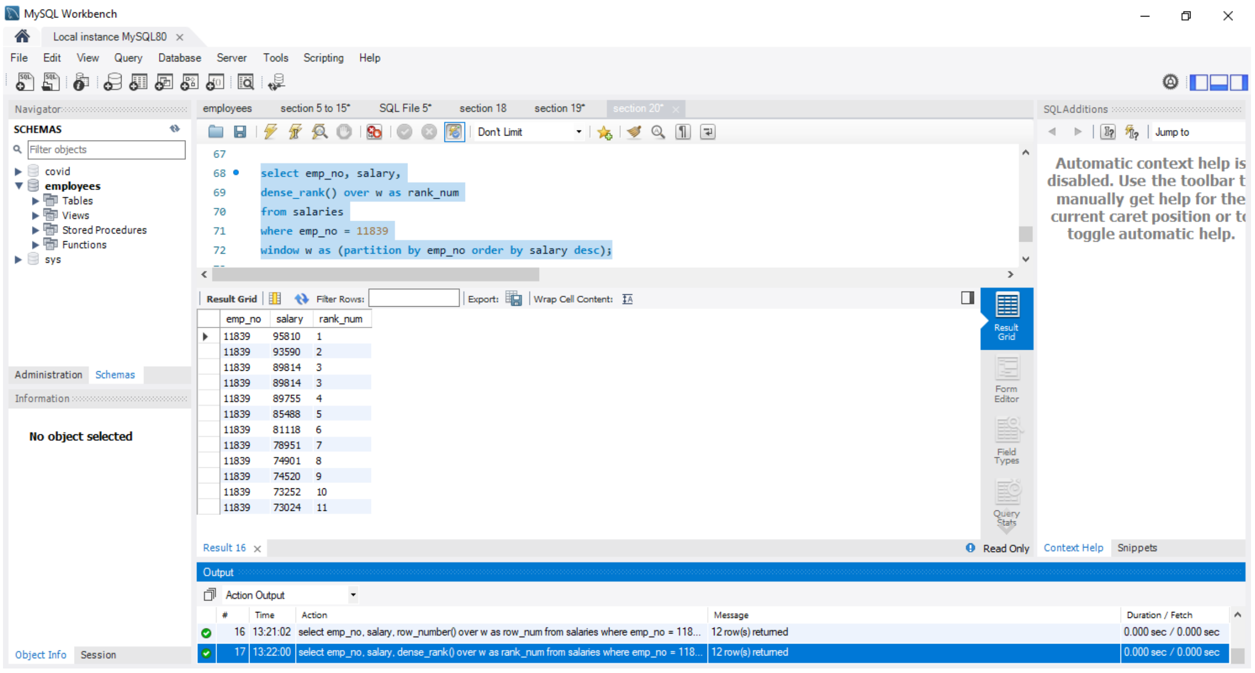
```
#   Time      Action                                                                                  Message               Duration / Fetch
15  13:20:50  SELECT a.emp_no, a.salary as min_salary FROM ( SELECT emp_no, salary, ROW_NUMB...       98976 row(s) returned  11.265 sec / 0.563 sec
16  13:21:02  select emp_no, salary, row_number() over w as row_num from salaries where emp_no = 118... 12 row(s) returned     0.000 sec / 0.000 sec
```



```
select emp_no, salary,
rank() over w as rank_num
from salaries
where emp_no = 11839
window w as (partition by emp_no order by salary desc);
```

emp_no	salary	rank_num
11839	95810	1
11839	93590	2
11839	89814	3
11839	89814	3
11839	89755	5
11839	85488	6
11839	81118	7
11839	78951	8
11839	74901	9
11839	74520	10
11839	73252	11
11839	73024	12

```
#  Time      Action                                                                                  Message             Duration / Fetch
3  13:17:53  select emp_no, salary, row_number() over w as row_num from salaries where emp_no = 1183... 12 row(s) returned  0.000 sec / 0.000 sec
4  13:18:18  select emp_no, salary, rank() over w as rank_num from salaries where emp_no = 11839 wind... 12 row(s) returned  0.109 sec / 0.000 sec
```



```sql
select emp_no, salary,
dense_rank() over w as rank_num
from salaries
where emp_no = 11839
window w as (partition by emp_no order by salary desc);
```

```
emp_no  salary  rank_num
11839   95810   1
11839   93590   2
11839   89814   3
11839   89814   3
11839   89755   4
11839   85488   5
11839   81118   6
11839   78951   7
11839   74901   8
11839   74520   9
11839   73252   10
11839   73024   11
```

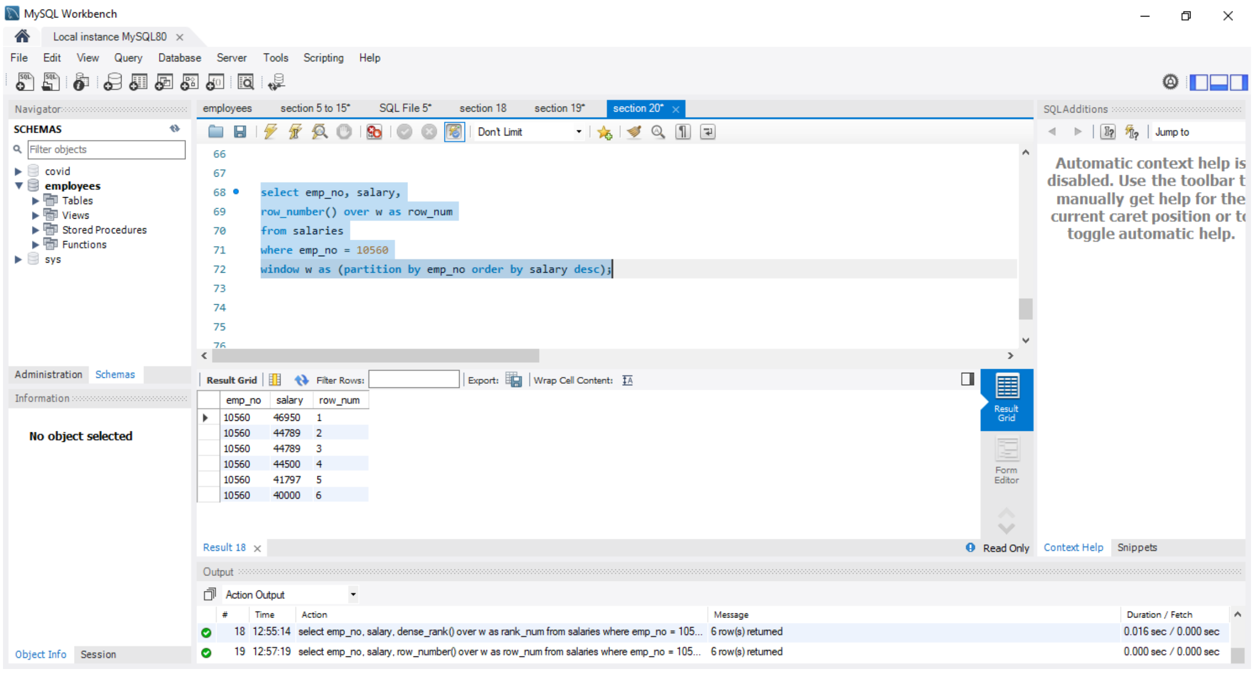
```
#   Time      Action                                                                                  Message              Duration / Fetch
16  13:21:02  select emp_no, salary, row_number() over w as row_num from salaries where emp_no = 118...  12 row(s) returned   0.000 sec / 0.000 sec
17  13:22:00  select emp_no, salary, dense_rank() over w as rank_num from salaries where emp_no = 118... 12 row(s) returned   0.000 sec / 0.000 sec
```

Exercise #1:

Write a query containing a window function to obtain all salary values that employee number 10560 has ever signed a contract for.

Order and display the obtained salary values from highest to lowest.

Solution:



```sql
select emp_no, salary,
row_number() over w as row_num
from salaries
where emp_no = 10560
window w as (partition by emp_no order by salary desc);
```

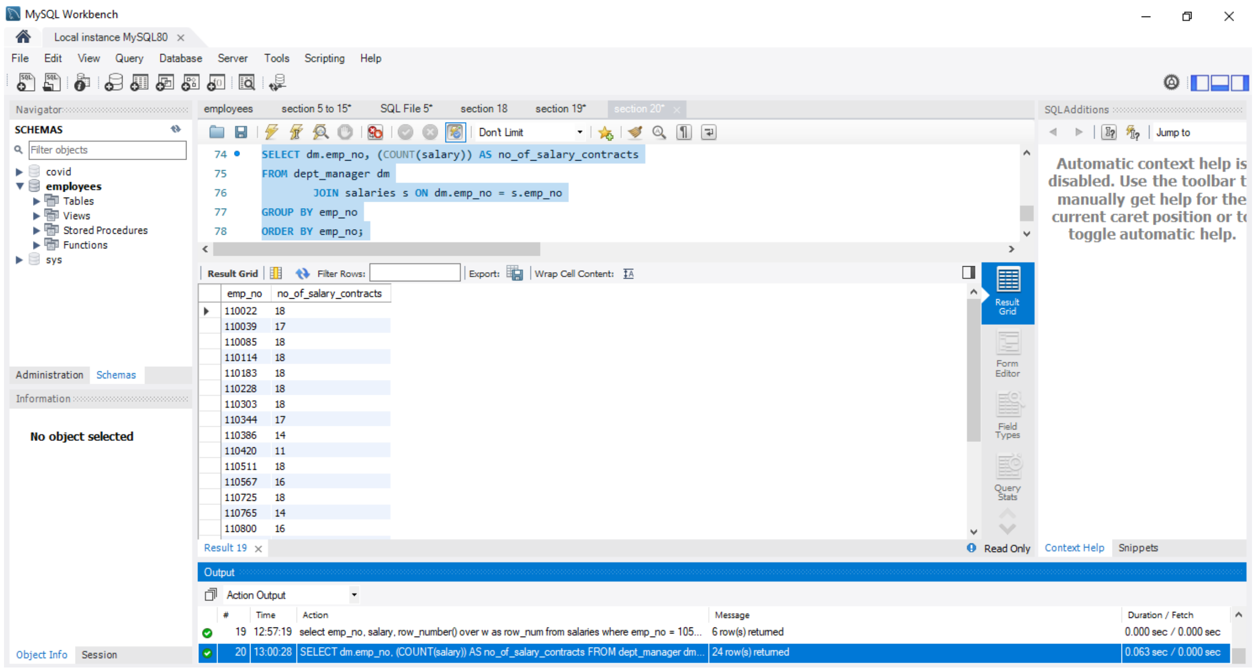
```
emp_no  salary  row_num
10560   46950   1
10560   44789   2
10560   44789   3
10560   44500   4
10560   41797   5
10560   40000   6
```

```
#   Time      Action                                                                                  Message             Duration / Fetch
18  12:55:14  select emp_no, salary, dense_rank() over w as rank_num from salaries where emp_no = 105... 6 row(s) returned   0.016 sec / 0.000 sec
19  12:57:19  select emp_no, salary, row_number() over w as row_num from salaries where emp_no = 105...  6 row(s) returned   0.000 sec / 0.000 sec
```

Exercise #2:

Write a query that upon execution, displays the number of salary contracts that each manager has ever signed while working in the company.

Solution:



```sql
SELECT dm.emp_no, (COUNT(salary)) AS no_of_salary_contracts
FROM dept_manager dm
        JOIN salaries s ON dm.emp_no = s.emp_no
GROUP BY emp_no
ORDER BY emp_no;
```

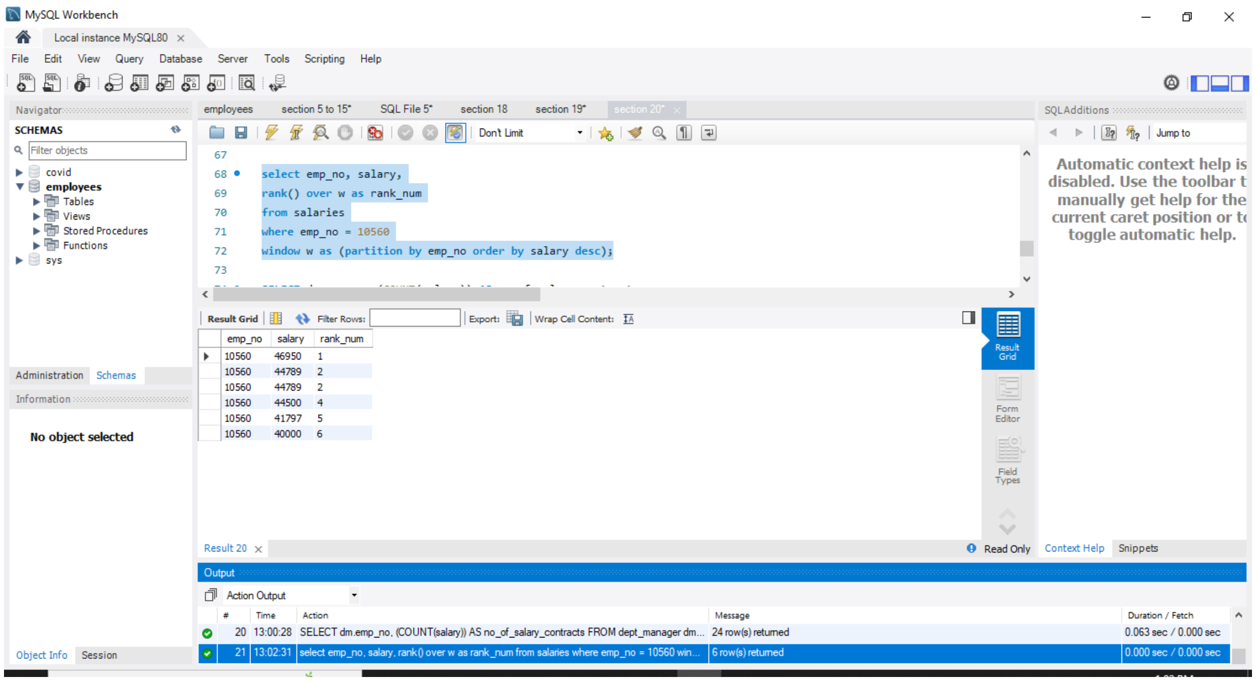
emp_no	no_of_salary_contracts
110022	18
110039	17
110085	18
110114	18
110183	18
110228	18
110303	18
110344	17
110386	14
110420	11
110511	18
110567	16
110725	18
110765	14
110800	16

```
#   Time      Action                                                                          Message             Duration / Fetch
19  12:57:19  select emp_no, salary, row_number() over w as row_num from salaries where emp_no = 105...  6 row(s) returned   0.000 sec / 0.000 sec
20  13:00:28  SELECT dm.emp_no, (COUNT(salary)) AS no_of_salary_contracts FROM dept_manager dm...  24 row(s) returned  0.063 sec / 0.000 sec
```

Exercise #3:

Write a query that upon execution retrieves a result set containing all salary values that employee 10560 has ever signed a contract for. Use a window function to rank all salary values from highest to lowest in a way that equal salary values bear the same rank and that gaps in the obtained ranks for subsequent rows are allowed.

Solution:



```sql
select emp_no, salary,
rank() over w as rank_num
from salaries
where emp_no = 10560
window w as (partition by emp_no order by salary desc);
```

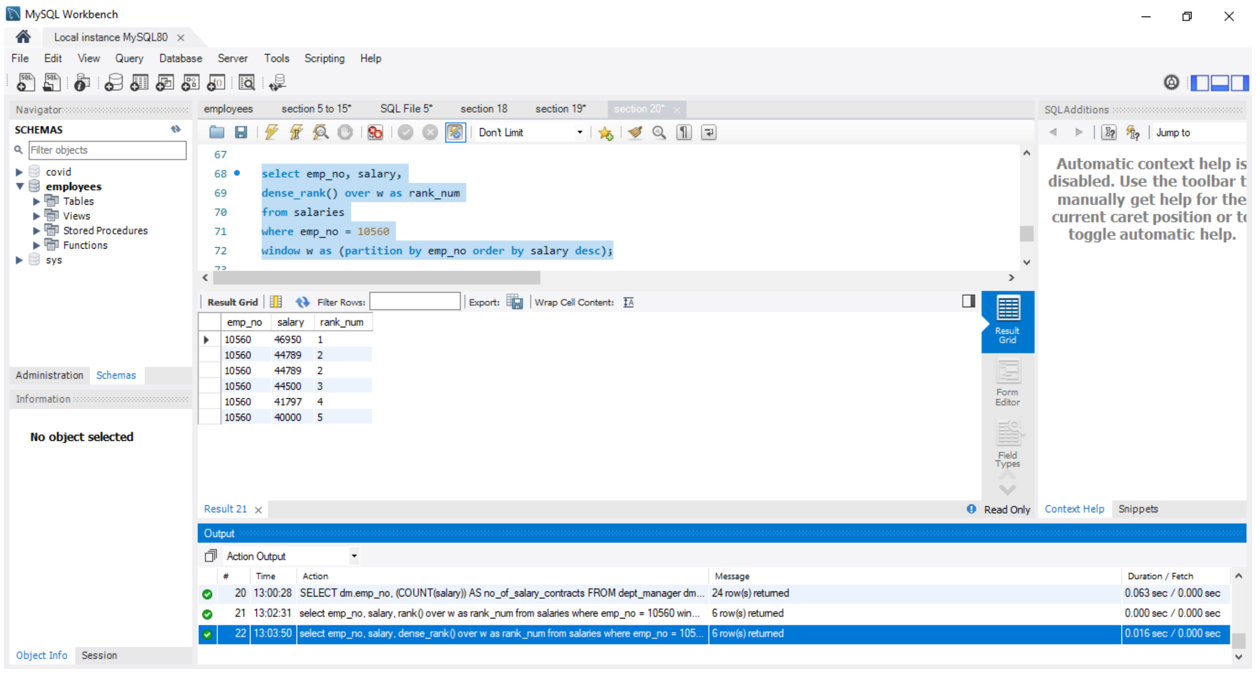
emp_no	salary	rank_num
10560	46950	1
10560	44789	2
10560	44789	2
10560	44500	4
10560	41797	5
10560	40000	6

```
#   Time      Action                                                                          Message            Duration / Fetch
20  13:00:28  SELECT dm.emp_no, (COUNT(salary)) AS no_of_salary_contracts FROM dept_manager dm...  24 row(s) returned  0.063 sec / 0.000 sec
21  13:02:31  select emp_no, salary, rank() over w as rank_num from salaries where emp_no = 10560 win...  6 row(s) returned  0.000 sec / 0.000 sec
```

Exercise #4:

Write a query that upon execution retrieves a result set containing all salary values that employee 10560 has ever signed a contract for. Use a window function to rank all salary values from highest to lowest in a way that equal salary values bear the same rank and that gaps in the obtained ranks for subsequent rows are not allowed.

Solution:



```sql
select emp_no, salary,
dense_rank() over w as rank_num
from salaries
where emp_no = 10560
window w as (partition by emp_no order by salary desc);
```

emp_no	salary	rank_num
10560	46950	1
10560	44789	2
10560	44789	2
10560	44500	3
10560	41797	4
10560	40000	5

```
#   Time      Action                                                                          Message            Duration / Fetch
20  13:00:28  SELECT dm.emp_no, (COUNT(salary)) AS no_of_salary_contracts FROM dept_manager dm...  24 row(s) returned  0.063 sec / 0.000 sec
21  13:02:31  select emp_no, salary, rank() over w as rank_num from salaries where emp_no = 10560 win...  6 row(s) returned  0.000 sec / 0.000 sec
22  13:03:50  select emp_no, salary, dense_rank() over w as rank_num from salaries where emp_no = 105...  6 row(s) returned  0.016 sec / 0.000 sec
```

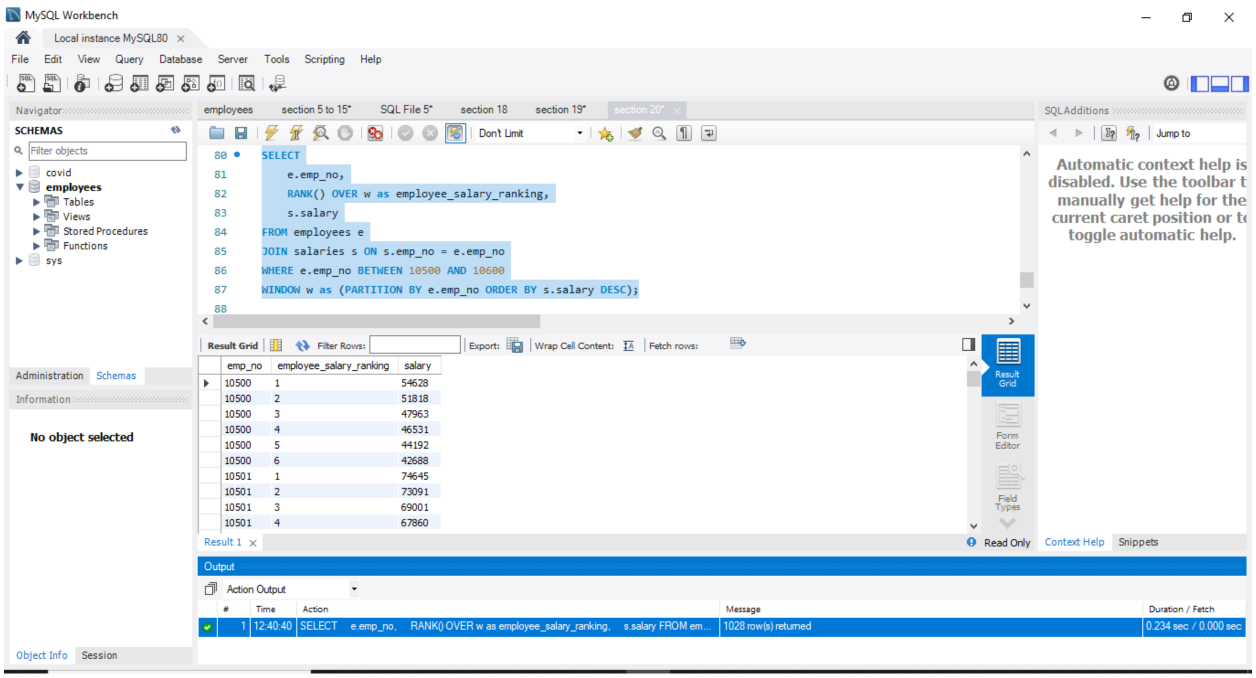
Working with MySQL Ranking Window Functions and Joins Together - Exercise

Exercise #1:

Write a query that ranks the salary values in descending order of all contracts signed by employees numbered between 10500 and 10600 inclusive. Let equal salary values for one and the same employee bear the same rank. Also, allow gaps in the ranks obtained for their subsequent rows.

Use a join on the "employees" and "salaries" tables to obtain the desired result.

Solution:



```sql
SELECT
    e.emp_no,
    RANK() OVER w as employee_salary_ranking,
    s.salary
FROM employees e
JOIN salaries s ON s.emp_no = e.emp_no
WHERE e.emp_no BETWEEN 10500 AND 10600
WINDOW w as (PARTITION BY e.emp_no ORDER BY s.salary DESC);
```

emp_no	employee_salary_ranking	salary
10500	1	54628
10500	2	51818
10500	3	47963
10500	4	46531
10500	5	44192
10500	6	42688
10501	1	74645
10501	2	73091
10501	3	69001
10501	4	67860

```
#  Time      Action                                                                        Message               Duration / Fetch
1  12:40:40  SELECT    e.emp_no,    RANK() OVER w as employee_salary_ranking,    s.salary FROM em...  1028 row(s) returned  0.234 sec / 0.000 sec
```

Exercise #2:

Write a query that ranks the salary values in descending order of the following contracts from the "employees" database:

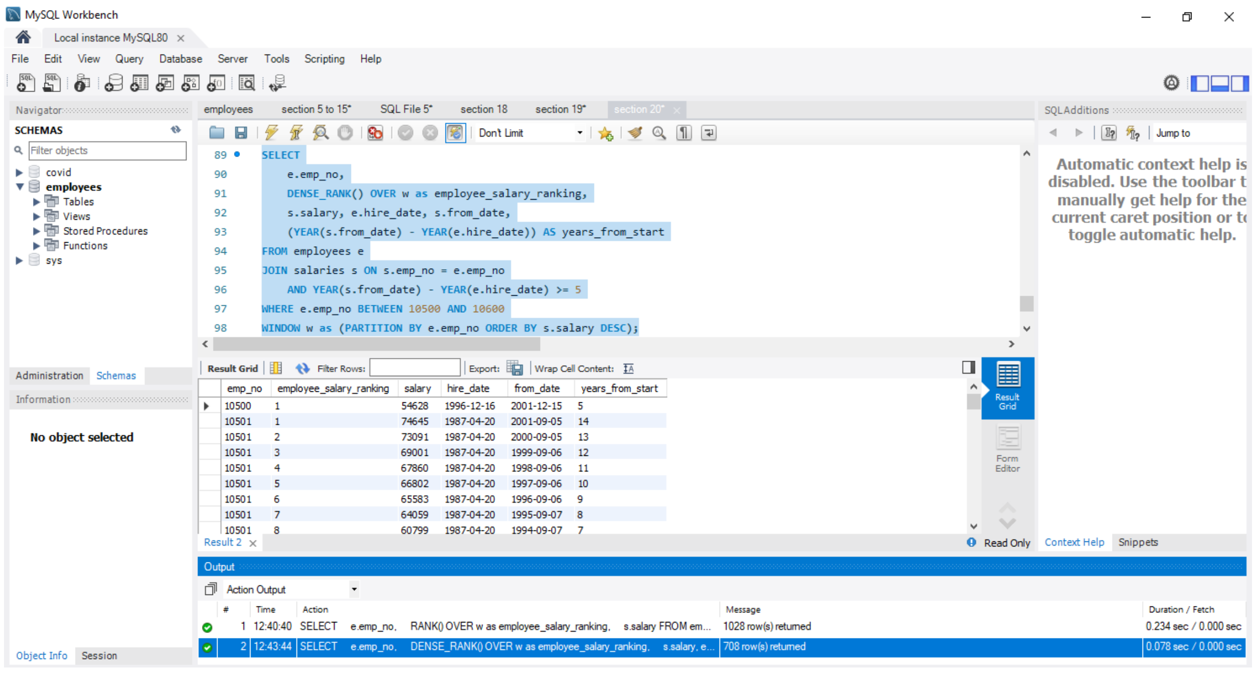
- contracts that have been signed by employees numbered between 10500 and 10600 inclusive.

- contracts that have been signed at least 4 full-years after the date when the given employee was hired in the company for the first time.

In addition, let equal salary values of a certain employee bear the same rank. Do not allow gaps in the ranks obtained for their subsequent rows.

Use a join on the "employees" and "salaries" tables to obtain the desired result.

Solution:



```sql
SELECT
    e.emp_no,
    DENSE_RANK() OVER w as employee_salary_ranking,
    s.salary, e.hire_date, s.from_date,
    (YEAR(s.from_date) - YEAR(e.hire_date)) AS years_from_start
FROM employees e
JOIN salaries s ON s.emp_no = e.emp_no
    AND YEAR(s.from_date) - YEAR(e.hire_date) >= 5
WHERE e.emp_no BETWEEN 10500 AND 10600
WINDOW w as (PARTITION BY e.emp_no ORDER BY s.salary DESC);
```

emp_no	employee_salary_ranking	salary	hire_date	from_date	years_from_start
10500	1	54628	1996-12-16	2001-12-15	5
10501	1	74645	1987-04-20	2001-09-05	14
10501	2	73091	1987-04-20	2000-09-05	13
10501	3	69001	1987-04-20	1999-09-06	12
10501	4	67860	1987-04-20	1998-09-06	11
10501	5	66802	1987-04-20	1997-09-06	10
10501	6	65583	1987-04-20	1996-09-06	9
10501	7	64059	1987-04-20	1995-09-07	8
10501	8	60799	1987-04-20	1994-09-07	7

```
#  Time      Action                                                                          Message              Duration / Fetch
1  12:40:40  SELECT  e.emp_no,  RANK() OVER w as employee_salary_ranking,  s.salary FROM em...  1028 row(s) returned  0.234 sec / 0.000 sec
2  12:43:44  SELECT  e.emp_no,  DENSE_RANK() OVER w as employee_salary_ranking,  s.salary, e...  708 row(s) returned  0.078 sec / 0.000 sec
```

The LAG () and LEAD () Value Window Functions - Exercise

Exercise #1:

Write a query that can extract the following information from the "employees" database:

- The salary values (in ascending order) of the contracts signed by all employees numbered between 10500 and 10600 inclusive

- A column showing the previous salary from the given ordered list

- A column showing the subsequent salary from the given ordered list

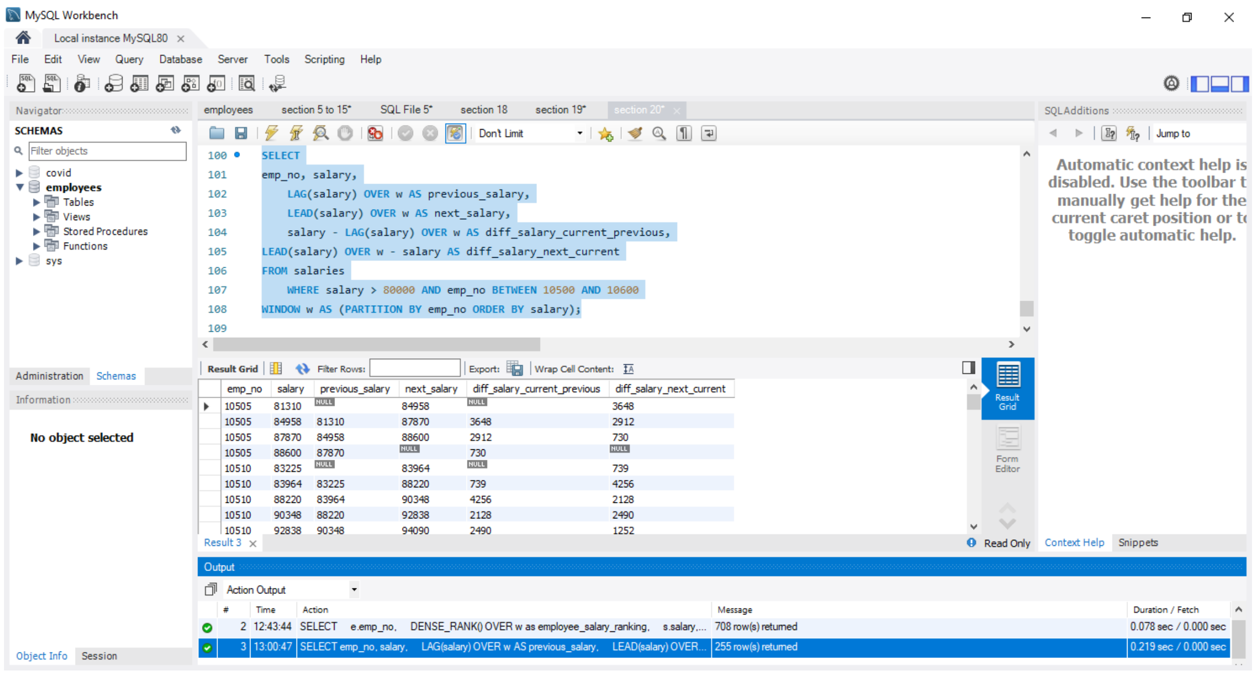
- A column displaying the difference between the current salary of a certain employee and their previous salary

- A column displaying the difference between the next salary of a certain employee and their current salary

Limit the output to salary values higher than $80,000 only.

Also, to obtain a meaningful result, partition the data by employee number.

Solution:



```sql
SELECT
emp_no, salary,
    LAG(salary) OVER w AS previous_salary,
    LEAD(salary) OVER w AS next_salary,
    salary - LAG(salary) OVER w AS diff_salary_current_previous,
LEAD(salary) OVER w - salary AS diff_salary_next_current
FROM salaries
    WHERE salary > 80000 AND emp_no BETWEEN 10500 AND 10600
WINDOW w AS (PARTITION BY emp_no ORDER BY salary);
```

emp_no	salary	previous_salary	next_salary	diff_salary_current_previous	diff_salary_next_current
10505	81310	NULL	84958	NULL	3648
10505	84958	81310	87870	3648	2912
10505	87870	84958	88600	2912	730
10505	88600	87870	NULL	730	NULL
10510	83225	NULL	83964	NULL	739
10510	83964	83225	88220	739	4256
10510	88220	83964	90348	4256	2128
10510	90348	88220	92838	2128	2490
10510	92838	90348	94090	2490	1252

	#	Time	Action	Message	Duration / Fetch
✓	2	12:43:44	SELECT e.emp_no, DENSE_RANK() OVER w as employee_salary_ranking, s.salary,...	708 row(s) returned	0.078 sec / 0.000 sec
✓	3	13:00:47	SELECT emp_no, salary, LAG(salary) OVER w AS previous_salary, LEAD(salary) OVER...	255 row(s) returned	0.219 sec / 0.000 sec

Exercise #2:

The MySQL LAG() and LEAD() value window functions can have a second argument, designating how many rows/steps back (for LAG()) or forth (for LEAD()) we'd like to refer to with respect to a given record.

With that in mind, create a query whose result set contains data arranged by the salary values associated to each employee number (in ascending order). Let the output contain the following six columns:

- The employee number

- The salary value of an employee's contract (i.e. which we'll consider as the employee's current salary)

- The employee's previous salary

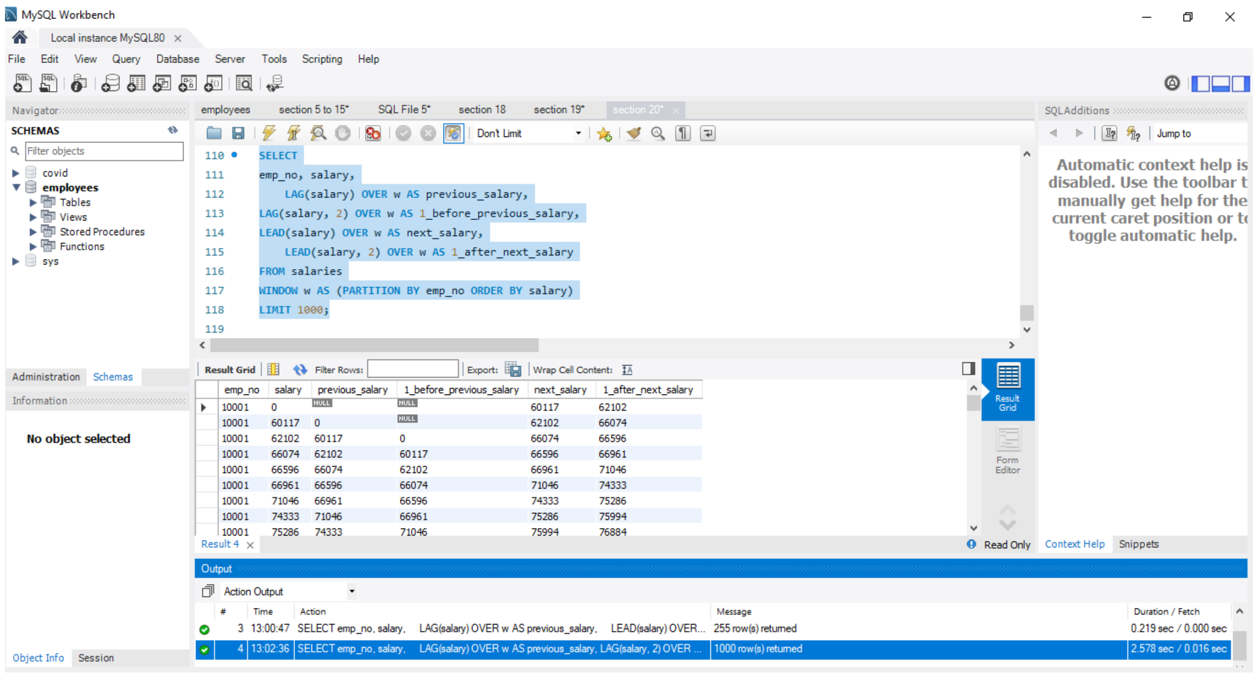
- The employee's contract salary value preceding their previous salary

- The employee's next salary

- The employee's contract salary value subsequent to their next salary

Restrict the output to the first 1000 records you can obtain.

Solution:



```sql
SELECT
emp_no, salary,
    LAG(salary) OVER w AS previous_salary,
LAG(salary, 2) OVER w AS 1_before_previous_salary,
LEAD(salary) OVER w AS next_salary,
    LEAD(salary, 2) OVER w AS 1_after_next_salary
FROM salaries
WINDOW w AS (PARTITION BY emp_no ORDER BY salary)
LIMIT 1000;
```

emp_no	salary	previous_salary	1_before_previous_salary	next_salary	1_after_next_salary
10001	0	NULL	NULL	60117	62102
10001	60117	0	NULL	62102	66074
10001	62102	60117	0	66074	66596
10001	66074	62102	60117	66596	66961
10001	66596	66074	62102	66961	71046
10001	66961	66596	66074	71046	74333
10001	71046	66961	66596	74333	75286
10001	74333	71046	66961	75286	75994
10001	75286	74333	71046	75994	76884

```
#  Time      Action                                                                        Message               Duration / Fetch
3  13:00:47  SELECT emp_no, salary,  LAG(salary) OVER w AS previous_salary,  LEAD(salary) OVER...  255 row(s) returned   0.219 sec / 0.000 sec
4  13:02:36  SELECT emp_no, salary,  LAG(salary) OVER w AS previous_salary, LAG(salary, 2) OVER ...  1000 row(s) returned  2.578 sec / 0.016 sec
```

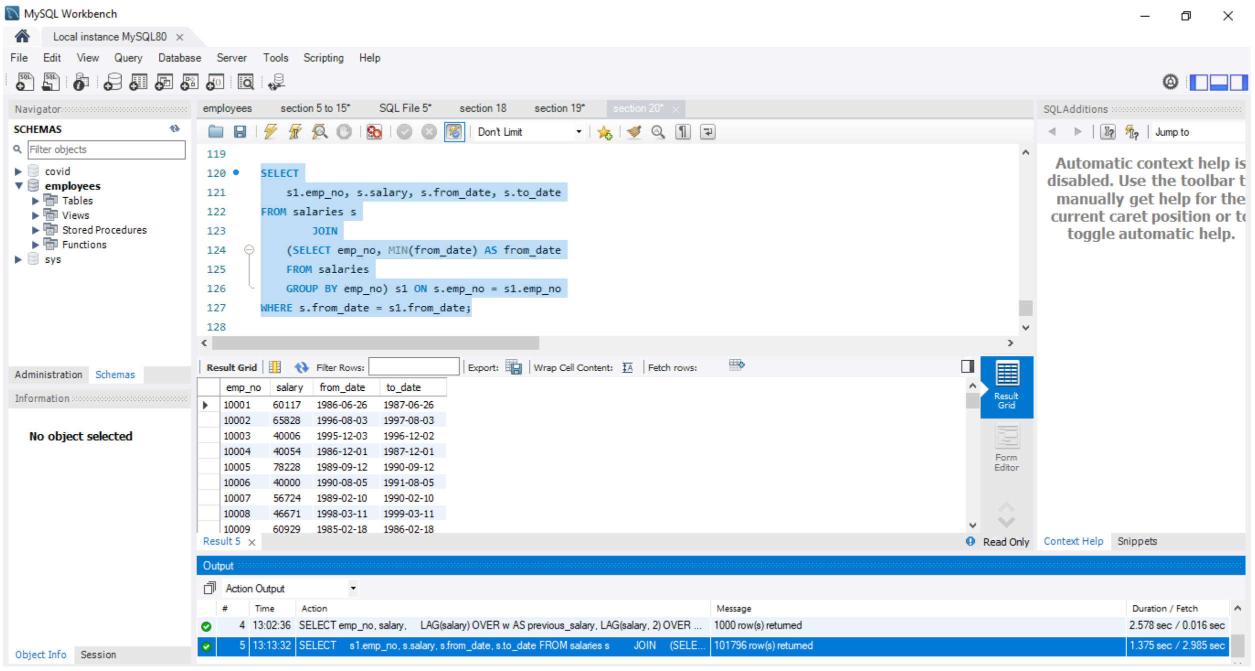
MySQL Aggregate Functions in the Context of Window Functions - Part I-Exercise

Exercise #1:

Create a query that upon execution returns a result set containing the employee numbers, contract salary values, start, and end dates of the first ever contracts that each employee signed for the company.

To obtain the desired output, refer to the data stored in the "salaries" table.

Solution:


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
covid
employees
Tables
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
No object selected
Object Info Session
employees section 5 to 15* SQL File 5* section 18 section 19* section 20*
Don't Limit
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Result Grid Filter Rows: Export: Wrap Cell Content: Fetch rows:
Result Grid
Form Editor
Result 5
Read Only
Context Help Snippets
Output
Action Output

```sql
SELECT
    s1.emp_no, s.salary, s.from_date, s.to_date
FROM salaries s
        JOIN
    (SELECT emp_no, MIN(from_date) AS from_date
    FROM salaries
    GROUP BY emp_no) s1 ON s.emp_no = s1.emp_no
WHERE s.from_date = s1.from_date;
```

emp_no	salary	from_date	to_date
10001	60117	1986-06-26	1987-06-26
10002	65828	1996-08-03	1997-08-03
10003	40006	1995-12-03	1996-12-02
10004	40054	1986-12-01	1987-12-01
10005	78228	1989-09-12	1990-09-12
10006	40000	1990-08-05	1991-08-05
10007	56724	1989-02-10	1990-02-10
10008	46671	1998-03-11	1999-03-11
10009	60929	1985-02-18	1986-02-18

#	Time	Action	Message	Duration / Fetch
4	13:02:36	SELECT emp_no, salary, LAG(salary) OVER w AS previous_salary, LAG(salary, 2) OVER ...	1000 row(s) returned	2.578 sec / 0.016 sec
5	13:13:32	SELECT s1.emp_no, s.salary, s.from_date, s.to_date FROM salaries s JOIN (SELE...	101796 row(s) returned	1.375 sec / 2.985 sec