Project:2

Title: Hands on project on SQL Window functions Using MySQL on employee's database of an organization.

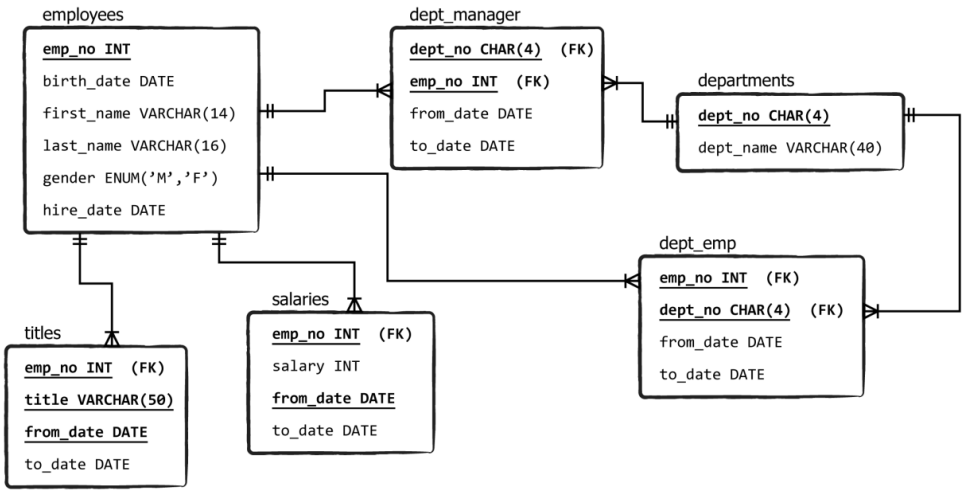
DONE BY:

JEWEL ALAM

Project Overview:

To analyze the hierarchy system of the organization by helping the human resource department of the organization. The system aims to provide actionable insights to improve decision-making and optimize human resource department.

Employees database structure:



Task: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.

</SQL>

select emp_no,dept_no,

Row_number() over (order by emp_no) as row_num

from

dept_manager;

Output:

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
110420	d004	10
110511	d005	11
110567	d005	12
110725	d006	13
110765	d006	14
110800	d006	15
110854	d006	16
111035	d007	17
111133	d007	18
111400	d008	19
111534	d008	20
111692	d009	21
111784	d009	22
111877	d009	23
111939	d009	24

Task: 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).

</SQL>

Select

emp_no,first_name,last_name,

row_number() over (partition by first_name order by last_name) as row_num

from

employees

limit 20;

Output:

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
283280	Aamer	Bahl	9
60922	Aamer	Bahl	10
405098	Aamer	Bahr	11
30404	Aamer	Basawa	12
419259	Aamer	Bashian	13
275611	Aamer	Bauknecht	14
466254	Aamer	Beausoleil	15
428971	Aamer	Belinskaya	16
94467	Aamer	Benantar	17
259555	Aamer	Bennis	18
296633	Aamer	Berendt	19
95079	Aamer	Beznosov	20

Task 3:

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.

</SQL>

```
select
 dm.emp_no,salary,
 Row_number() over(partition by emp_no order by salary asc) as row_1_salary_asc,
 row_number()over(partition by emp_no order by salary desc) as row_2_salary_desc
 from
 dept_manager dm
 join
 salaries s on dm.emp_no=s.emp_no
 limit 25;
```

Output:

emp_no	salary	row_1_salary_asc	row_2_salary_desc
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
110022	89204	9	10
110022	86797	8	11
110022	82871	7	12
110022	81784	6	13

110022	78443	5	14
110022	76211	4	15
110022	72970	3	16
110022	71820	2	17
110022	71166	1	18
110039	106491	17	1
110039	104115	16	2
110039	101901	15	3
110039	100350	14	4
110039	95993	13	5
110039	94250	12	6
110039	92469	11	7

Task 4:

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.

</SQL>

select dm.emp_no,salary,

row_number()over (partition by emp_no order by salary) 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

limit 20;

OUTPUT:

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
110022	89204	9	10
110022	86797	8	11
110022	82871	7	12
110022	81784	6	13
110022	78443	5	14
110022	76211	4	15
110022	72970	3	16
110022	71820	2	17
110022	71166	1	18
110039	106491	17	1
110039	104115	16	2

Task: 5

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.*

</SQL>

```
select emp_no,first_name ,
 row_number() over w as row_num1
 from employees
 window w as (partition by first_name order by emp_no asc)
 limit 20;
```

Output:

emp_no	first_name	row_num1
11800	Aamer	1
11935	Aamer	2
12160	Aamer	3
13011	Aamer	4
15332	Aamer	5
20678	Aamer	6
22279	Aamer	7
23269	Aamer	8
24404	Aamer	9
28043	Aamer	10
28162	Aamer	11
29217	Aamer	12
29981	Aamer	13
30404	Aamer	14
30710	Aamer	15
31646	Aamer	16

32854	Aamer	17
33326	Aamer	18
34878	Aamer	19
36146	Aamer	20

Task :6

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.

</SQL>

select

emp_no,salary,

row_number() over w as salary_row_number

from

salaries

where emp_no=10560

window w as (partition by emp_no order by salary desc);

Output:

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

Task: 7

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

```
select
dm.emp_no ,count(s.salary) as no_of_contracts
from
dept_manager dm
join
salaries s on dm.emp_no=s.emp_no
group by emp_no
order by emp_no;
```

Output:

emp_no	no_of_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
110854	14
111035	18
111133	16
111400	18
111534	15
111692	18

111784	15
111877	11
111939	14

Task:8

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.

</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);

Output:

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

Task: 9

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.

</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);

Output:

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

Task:10

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.

</SQL>

```
select
e.emp_no,
rank() over w as employees_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);
```

Output:

emp_no	employees_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
10501	5	66802
10501	6	65583
10501	7	64059
10501	8	60799
10501	9	60652
10501	10	60308

10501	11	60090
10501	12	58242
10501	13	56761
10501	14	53922

Task:11

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.

</SQL>

select

e.emp_no,

dense_rank() over was employee_salary_ranking,

s.salary,

e.hire_date,

(year(s.from_date)-year(e.hire_date)) as year_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 was as (partition by e.emp_no order by s.salary desc)

limit 20;

Output:

emp_no	employee_salary_ranking	salary	hire_date	year_from_start
10500	1	54628	16-12-1996	5
10501	1	74645	20-04-1987	14
10501	2	73091	20-04-1987	13
10501	3	69001	20-04-1987	12
10501	4	67860	20-04-1987	11
10501	5	66802	20-04-1987	10
10501	6	65583	20-04-1987	9
10501	7	64059	20-04-1987	8
10501	8	60799	20-04-1987	7
10501	9	60652	20-04-1987	6
10501	10	60308	20-04-1987	5
10502	1	60033	04-03-1993	8
10502	2	59636	04-03-1993	9
10502	3	56287	04-03-1993	7
10502	4	53992	04-03-1993	6
10502	5	50313	04-03-1993	5
10503	1	55115	30-04-1991	11
10503	2	51637	30-04-1991	10
10503	3	48243	30-04-1991	9
10503	4	45292	30-04-1991	8

Task :12

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.

</SQL>

```
select
salary,
lag(salary) over w as previous_salary_of_current,
lead(salary) over w as next_salary_of_current,
salary-lag(salary) over w as diff_previous_current,
lead(salary) over w -salary as diff_next_current
from
salaries
where salary>80000 and
emp_no between 10500 and 10600
window  w as (partition by emp_no order by salary)
limit 20 ;
```

Output:

salary	previous_salary_of_current	next_salary_of_current	diff_previous_current	diff_next_current
81310	NULL	84958	NULL	3648
84958	81310	87870	3648	2912
87870	84958	88600	2912	730
88600	87870	NULL	730	NULL
83225	NULL	83964	NULL	739
83964	83225	88220	739	4256
88220	83964	90348	4256	2128
90348	88220	92838	2128	2490
92838	90348	94090	2490	1252
94090	92838	98279	1252	4189
98279	94090	NULL	4189	NULL
86392	NULL	86680	NULL	288
86680	86392	87896	288	1216
87896	86680	91107	1216	3211
91107	87896	93189	3211	2082
93189	91107	96779	2082	3590
96779	93189	97107	3590	328
97107	96779	97250	328	143
97250	97107	100751	143	3501
100751	97250	102025	3501	1274

Task:13

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.

</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
limit 20;
```

Output:

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

10009	60929	18-02-1985	18-02-1986
10010	72488	24-11-1996	24-11-1997
10011	42365	22-01-1990	22-01-1991
10012	40000	18-12-1992	18-12-1993
10013	40000	20-10-1985	20-10-1986
10014	46168	29-12-1993	29-12-1994
10015	40000	19-09-1992	22-08-1993
10016	70889	11-02-1998	11-02-1999
10017	71380	03-08-1993	03-08-1994
10018	55881	03-04-1987	02-04-1988
10019	44276	30-04-1999	29-04-2000
10020	40000	30-12-1997	30-12-1998

Task: 14

Consider the employees' contracts that have been signed after the 1st of January 2000 and terminated before the 1st of January 2002 (as registered in the "dept_emp" table).

Create a MySQL query that will extract the following information about these employees:

- Their employee number

- The salary values of the latest contracts they have signed during the suggested time period

- The department they have been working in (as specified in the latest contract they've signed during the suggested time period)

- Use a window function to create a fourth field containing the average salary paid in the department the employee was last working in during the suggested time period. Name that field "average_salary_per_department".

Note1: This exercise is not related neither to the query you created nor to the output you obtained while solving the exercises after the previous lecture.

Note2: Now we are asking you to practically create the same query as the one we worked on during the video lecture; the only difference being to refer to contracts that have been valid within the period between the 1st of January 2000 and the 1st of January 2002.

Note3: We invite you solve this task after assuming that the "to_date" values stored in the "salaries" and "dept_emp" tables are greater than the "from_date" values stored in these same tables. If you doubt that, you could include a couple of lines in your code to ensure that this is the case anyway!

Hint: If you've worked correctly, you should obtain an output containing 200 rows.

</SQL>

```
SELECT
de2.emp_no, d.dept_name, s2.salary, AVG(s2.salary) OVER w AS
average_salary_each_department
 FROM
(SELECT
 de.emp_no, de.dept_no, de.from_date, de.to_date
 FROM
dept_emp de
 JOIN
(SELECT
 emp_no, MAX(from_date) AS from_date
 FROM
 dept_emp
 GROUP BY emp_no) de1 ON de1.emp_no = de.emp_no
 WHERE
 de.to_date < '2002-01-01'
 AND de.from_date > '2000-01-01'
 AND de.from_date = de1.from_date) de2
```

```
 JOIN
(SELECT
 s1.emp_no, s.salary, s.from_date, s.to_date
 FROM
salaries s
JOIN
(SELECT
emp_no, MAX(from_date) AS from_date
 FROM
 salaries
 GROUP BY emp_no) s1 ON s.emp_no = s1.emp_no
 WHERE
  s.to_date < '2002-01-01'
 AND s.from_date > '2000-01-01'

AND s.from_date = s1.from_date) s2 ON s2.emp_no = de2.emp_no
 JOIN
 departments d ON d.dept_no = de2.dept_no
 GROUP BY de2.emp_no, d.dept_name
 WINDOW w AS (PARTITION BY de2.dept_no)
 ORDER BY de2.emp_no, salary
limit 20;
```

Output:

emp_no	dept_name	salary	average_salary_per_department
11609	Marketing	75216	68583.5333
12693	Research	71443	53997.8519
12721	Quality Management	83814	52452.6842
14378	Production	53703	52520.1064
14615	Production	43614	52520.1064
14889	Sales	54033	65564.4211
14957	Sales	75779	65564.4211
15037	Research	51679	53997.8519
15334	Development	42250	50687.3125
15439	Quality Management	74545	52452.6842
15515	Production	43884	52520.1064
15527	Production	40659	52520.1064
16502	Customer Service	84607	61459.5161
16547	Production	47818	52520.1064
17250	Development	50293	50687.3125
17801	Sales	71569	65564.4211
18178	Development	40179	50687.3125
18332	Development	50121	50687.3125
19098	Customer Service	43167	61459.5161
19196	Marketing	66890	68583.5333

Conclusion : All the task related to window function of SQL were performed.