Window Functions

ISM 6218

Due on October 1st

The Avengers Team

"We will avenge every problem on our way"

Aitemir Yeskenov (Team Lead)

Nagarjuna Kanneganti

Sai Suraj Argula

Vinay Kumar Reddy Baradi

Table of Contents

Business Process Supported

For this assignment, we decided to use the Employee database where we created 5 columns to server the purpose of the assignment. We have used multiple functions to get the tailored results which accord with business needs. This document emphasizes more on using more advanced functions and operators specified.

Requirements Described

Run a series of experiments demonstrating:

- Over clause and Partition by
- Ranking
- Value
- Aggregate
- Compare to Group by
- Compare without the use of over clause

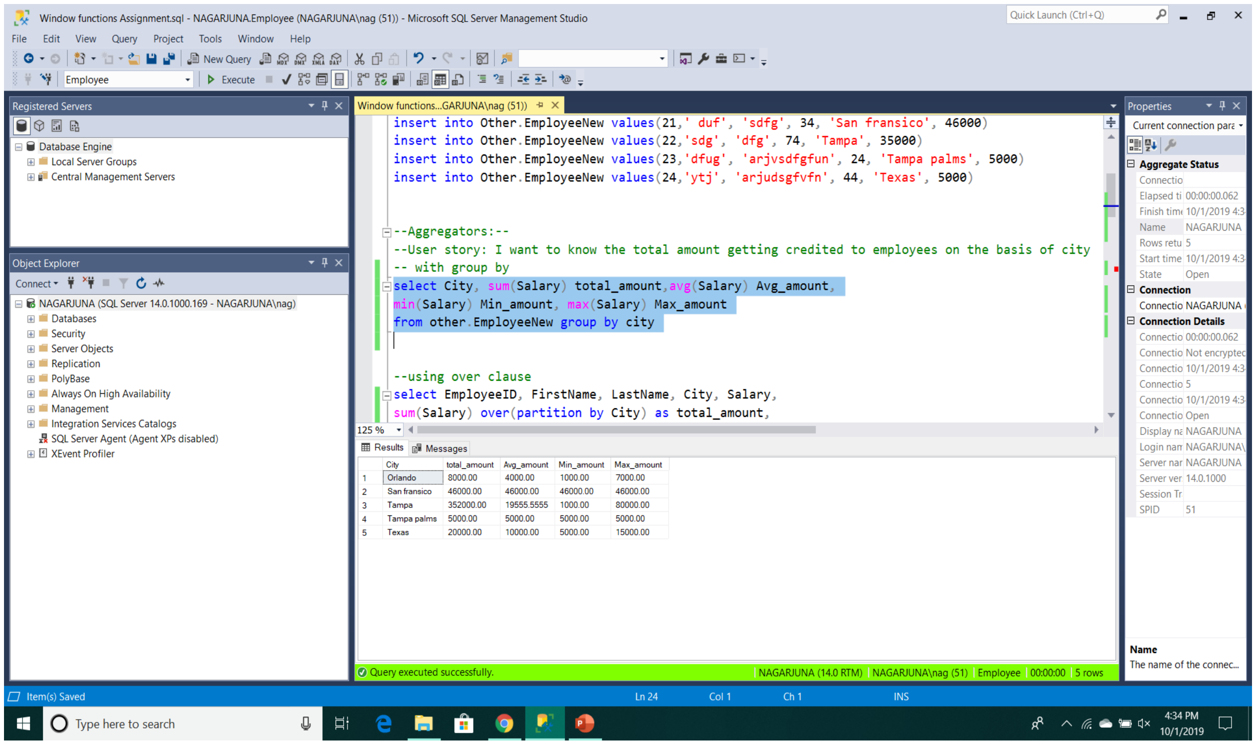
Must support a user story for each run.

You need to identify the specific requirements derived from the user story mapped to the clause or operator implemented.

Window aggregators

For this assignment, we have taken 4 different aggregators which are sum, avg, min and max for demonstration.

User story: In our company, In which city, who all are getting the highest salary, lowest salary, average salary metric and total amount needed for salaries for our employees on the basis of city in united states.



Window functions Assignment.sql - NAGARJUNA.Employee (NAGARJUNA\nag (51)) - Microsoft SQL Server Management Studio

```sql
insert into Other.EmployeeNew values(21,' duf', 'sdfg', 34, 'San fransico', 46000)
insert into Other.EmployeeNew values(22,'sdg', 'dfg', 74, 'Tampa', 35000)
insert into Other.EmployeeNew values(23,'dfug', 'arjvsdfgfun', 24, 'Tampa palms', 5000)
insert into Other.EmployeeNew values(24,'ytj', 'arjudsgfvfn', 44, 'Texas', 5000)


--Aggregators:--
--User story: I want to know the total amount getting credited to employees on the basis of city
-- with group by
select City, sum(Salary) total_amount,avg(Salary) Avg_amount,
min(Salary) Min_amount, max(Salary) Max_amount
from other.EmployeeNew group by city


--using over clause
select EmployeeID, FirstName, LastName, City, Salary,
sum(Salary) over(partition by City) as total_amount,
```

	City	total_amount	Avg_amount	Min_amount	Max_amount
1	Orlando	8000.00	4000.00	1000.00	7000.00
2	San fransico	46000.00	46000.00	46000.00	46000.00
3	Tampa	352000.00	19555.5555	1000.00	80000.00
4	Tampa palms	5000.00	5000.00	5000.00	5000.00
5	Texas	20000.00	10000.00	5000.00	15000.00

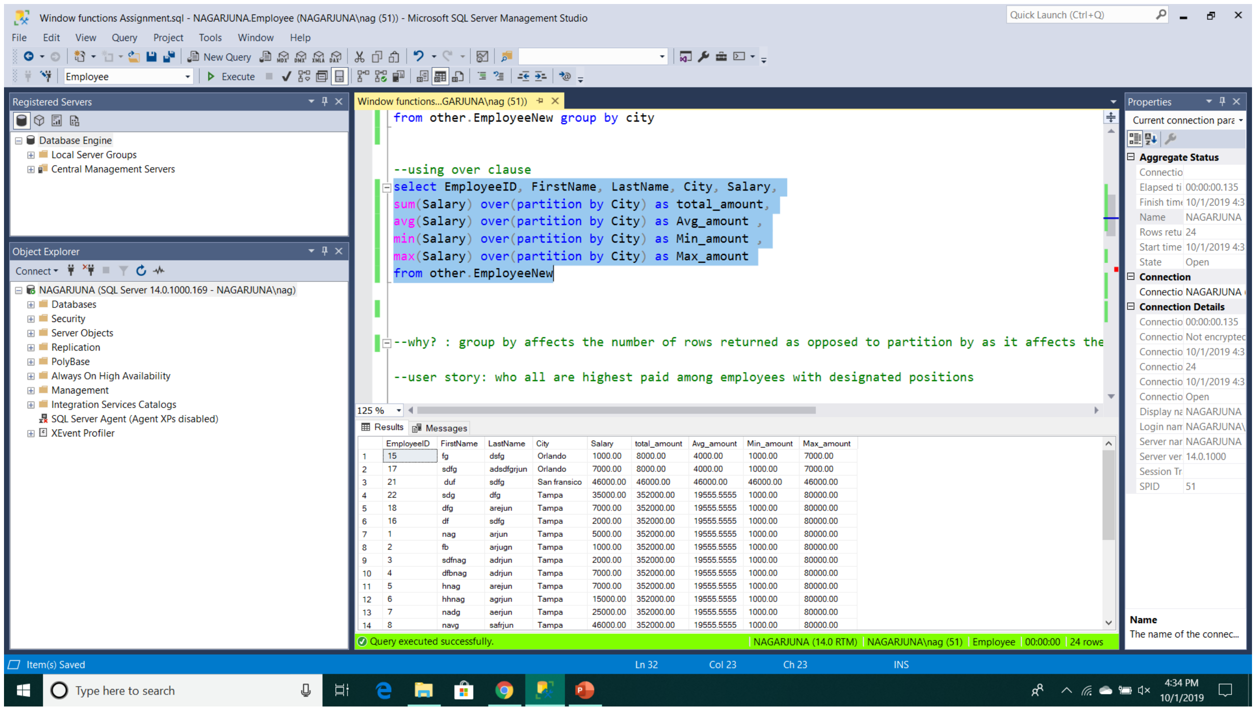
Query executed successfully. NAGARJUNA (14.0 RTM) NAGARJUNA\nag (51) Employee 00:00:00 5 rows

In the screenshot posted above, we are displaying the salaries of highest, lowest employee and an average salary for a person and total amount of money that required to pay for employees in city wise.

Why we need over clause and partition by? → group by affects the number of rows returned as opposed to partition by as it affects the windows of rows

Using of over clause and partition by:

To get all the details of employees we have used over clause and partition by for records according to cities because aggregates need group by clause and it only give results based on the group by clause.



```sql
from other.EmployeeNew group by city


--using over clause
select EmployeeID, FirstName, LastName, City, Salary,
sum(Salary) over(partition by City) as total_amount,
avg(Salary) over(partition by City) as Avg_amount ,
min(Salary) over(partition by City) as Min_amount ,
max(Salary) over(partition by City) as Max_amount
from other.EmployeeNew


--why? : group by affects the number of rows returned as opposed to partition by as it affects the

--user story: who all are highest paid among employees with designated positions
```

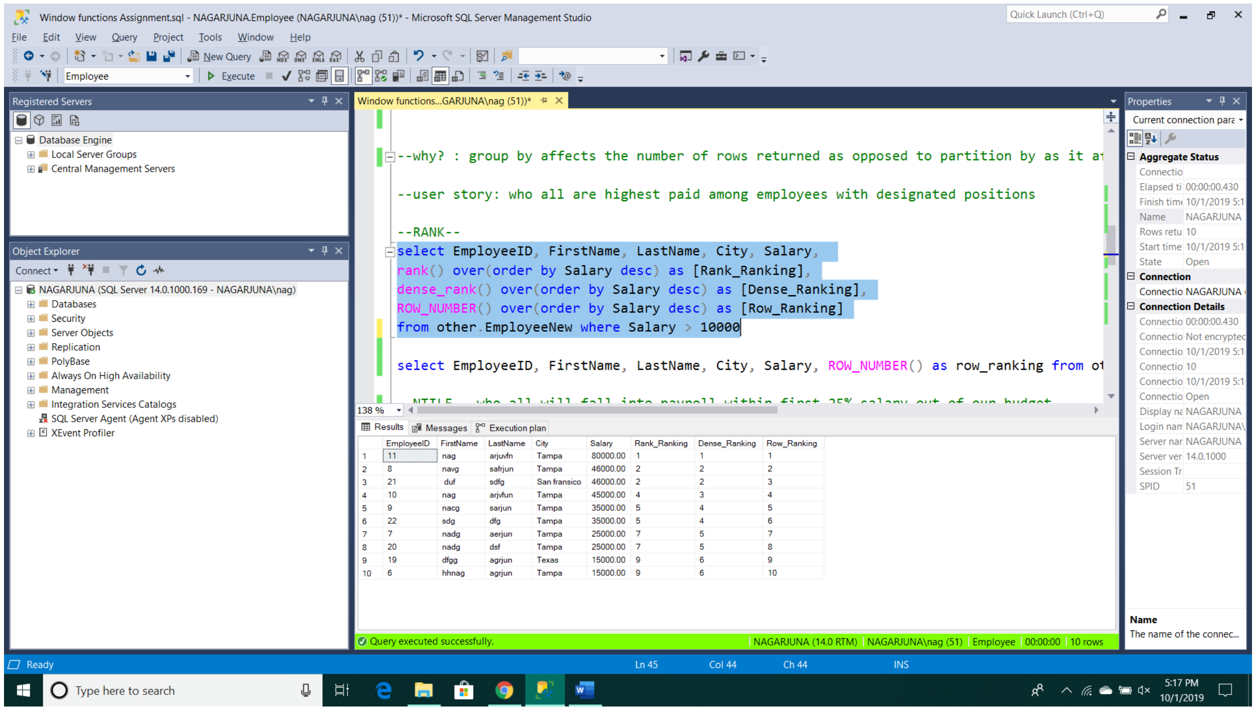
	EmployeeID	FirstName	LastName	City	Salary	total_amount	Avg_amount	Min_amount	Max_amount
1	15	fg	dsfg	Orlando	1000.00	8000.00	4000.00	1000.00	7000.00
2	17	sdfg	adsdfgrjun	Orlando	7000.00	8000.00	4000.00	1000.00	7000.00
3	21	duf	sdfg	San fransico	46000.00	46000.00	46000.00	46000.00	46000.00
4	22	sdg	dfg	Tampa	35000.00	352000.00	19555.5555	1000.00	80000.00
5	18	dfg	arejun	Tampa	7000.00	352000.00	19555.5555	1000.00	80000.00
6	16	df	sdfg	Tampa	2000.00	352000.00	19555.5555	1000.00	80000.00
7	1	nag	arjun	Tampa	5000.00	352000.00	19555.5555	1000.00	80000.00
8	2	fb	arjugn	Tampa	1000.00	352000.00	19555.5555	1000.00	80000.00
9	3	sdfnag	adrjun	Tampa	2000.00	352000.00	19555.5555	1000.00	80000.00
10	4	dfbnag	adrjun	Tampa	7000.00	352000.00	19555.5555	1000.00	80000.00
11	5	hnag	arejun	Tampa	7000.00	352000.00	19555.5555	1000.00	80000.00
12	6	hhnag	agrjun	Tampa	15000.00	352000.00	19555.5555	1000.00	80000.00
13	7	nadg	aerjun	Tampa	25000.00	352000.00	19555.5555	1000.00	80000.00
14	8	navg	safrjun	Tampa	46000.00	352000.00	19555.5555	1000.00	80000.00

Query executed successfully.

In this screenshot, we can display every employee detail and also the salaries of them according to cities using aggregate functions with over and partition by clause.

Ranking functions

User story: In our company, who all employees are paid more than 10000$ per annum with their designated ranking.



```
--why? : group by affects the number of rows returned as opposed to partition by as it a

--user story: who all are highest paid among employees with designated positions

--RANK--
select EmployeeID, FirstName, LastName, City, Salary,
rank() over(order by Salary desc) as [Rank_Ranking],
dense_rank() over(order by Salary desc) as [Dense_Ranking],
ROW_NUMBER() over(order by Salary desc) as [Row_Ranking]
from other.EmployeeNew where Salary > 10000

select EmployeeID, FirstName, LastName, City, Salary, ROW_NUMBER() as row_ranking from o
```

	EmployeeID	FirstName	LastName	City	Salary	Rank_Ranking	Dense_Ranking	Row_Ranking
1	11	nag	arjuvfn	Tampa	80000.00	1	1	1
2	8	navg	safrjun	Tampa	46000.00	2	2	2
3	21	duf	sdfg	San fransico	46000.00	2	2	3
4	10	nag	arjvfun	Tampa	45000.00	4	3	4
5	9	nacg	sarjun	Tampa	35000.00	5	4	5
6	22	sdg	dfg	Tampa	35000.00	5	4	6
7	7	nadg	aerjun	Tampa	25000.00	7	5	7
8	20	nadg	dsf	Tampa	25000.00	7	5	8
9	19	dfgg	agrjun	Texas	15000.00	9	6	9
10	6	hhnag	agrjun	Tampa	15000.00	9	6	10

Query executed successfully.

In screenshot, We have used rank, dense rank and row number to rank the employees who are paid more than 10000$ to know their position in the company.

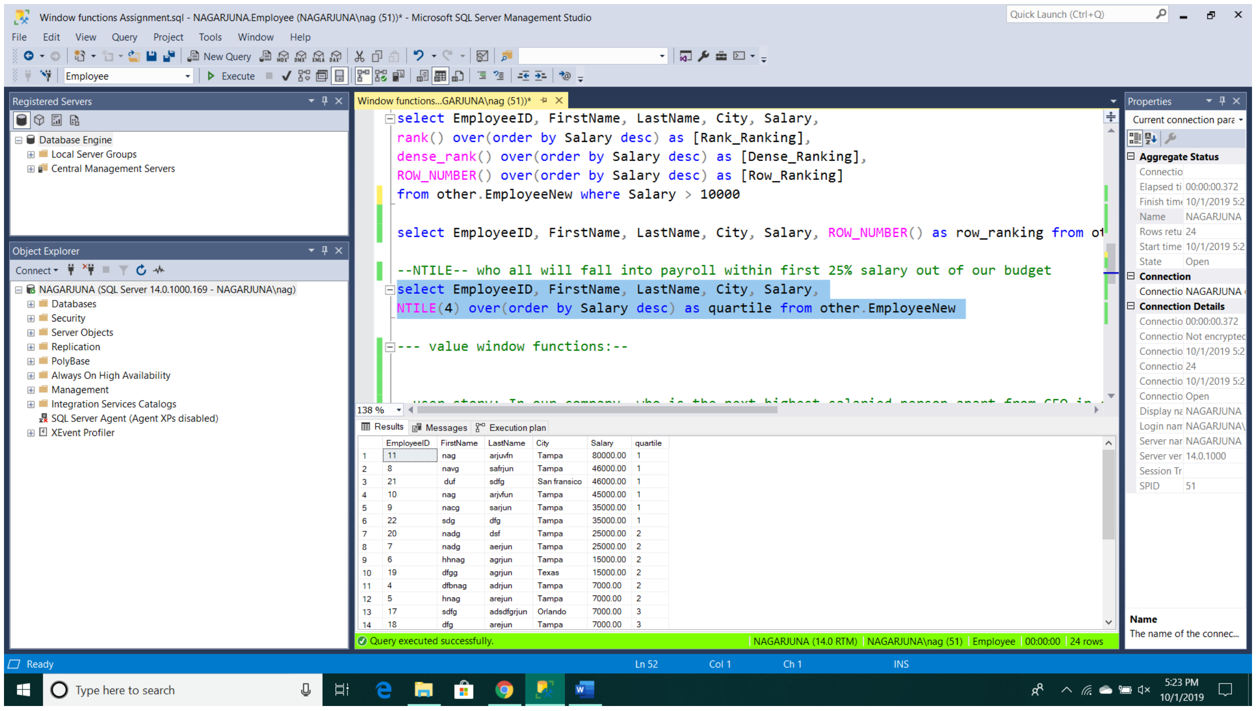
Rank is used to know where they stand even if some employees are earning same amount of salary.

Dense rank is used to know correct ranking for the employees

Row number is used to know the number of employees who are in that bucket (salary greater than 10000).

NTILE function:

User story: In our company, who all will fall into the first 25% payroll



```sql
select EmployeeID, FirstName, LastName, City, Salary,
rank() over(order by Salary desc) as [Rank_Ranking],
dense_rank() over(order by Salary desc) as [Dense_Ranking],
ROW_NUMBER() over(order by Salary desc) as [Row_Ranking]
from other.EmployeeNew where Salary > 10000

select EmployeeID, FirstName, LastName, City, Salary, ROW_NUMBER() as row_ranking from ot

--NTILE-- who all will fall into payroll within first 25% salary out of our budget
select EmployeeID, FirstName, LastName, City, Salary,
NTILE(4) over(order by Salary desc) as quartile from other.EmployeeNew

--- value window functions:--
```

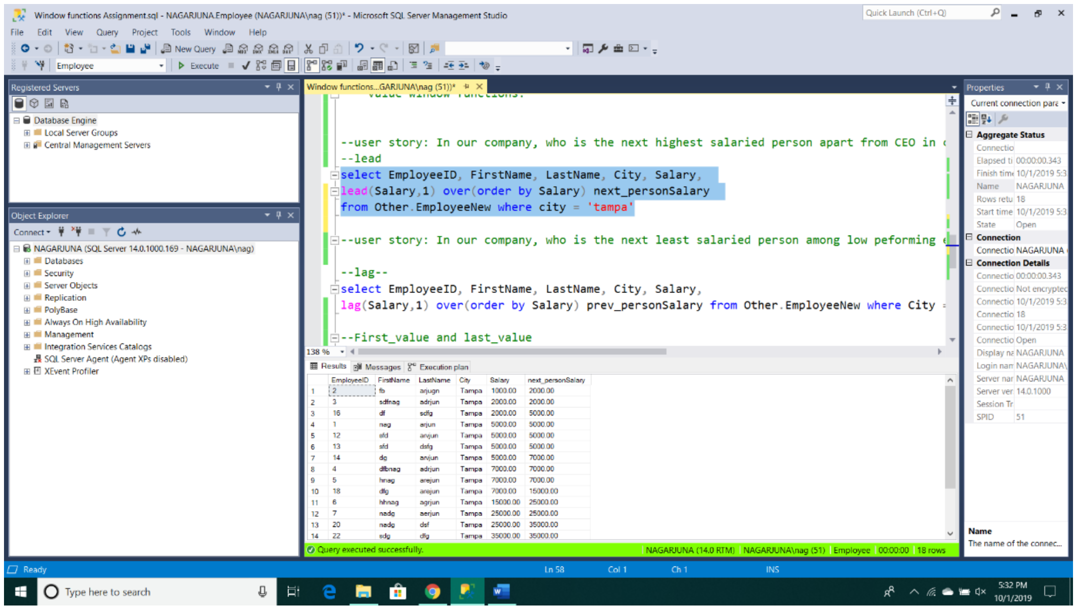
	EmployeeID	FirstName	LastName	City	Salary	quartile
1	11	nag	arjuvfn	Tampa	80000.00	1
2	8	navg	safrjun	Tampa	46000.00	1
3	21	duf	sdfg	San fransico	46000.00	1
4	10	nag	arjvfun	Tampa	45000.00	1
5	9	nacg	sarjun	Tampa	35000.00	1
6	22	sdg	dfg	Tampa	35000.00	1
7	20	nadg	dsf	Tampa	25000.00	2
8	7	nadg	aerjun	Tampa	25000.00	2
9	6	hhnag	agrjun	Tampa	15000.00	2
10	19	dfgg	agrjun	Texas	15000.00	2
11	4	dfbnag	adrjun	Tampa	7000.00	2
12	5	hnag	arejun	Tampa	7000.00	2
13	17	sdfg	adsdfgrjun	Orlando	7000.00	3
14	18	dfg	arejun	Tampa	7000.00	3

Query executed successfully.

In this screenshot, we can see that the highest salaried persons till 30000 falls into 25% payroll.

Window value functions

User story: In our company, who all are highest salaried persons apart from others in our city



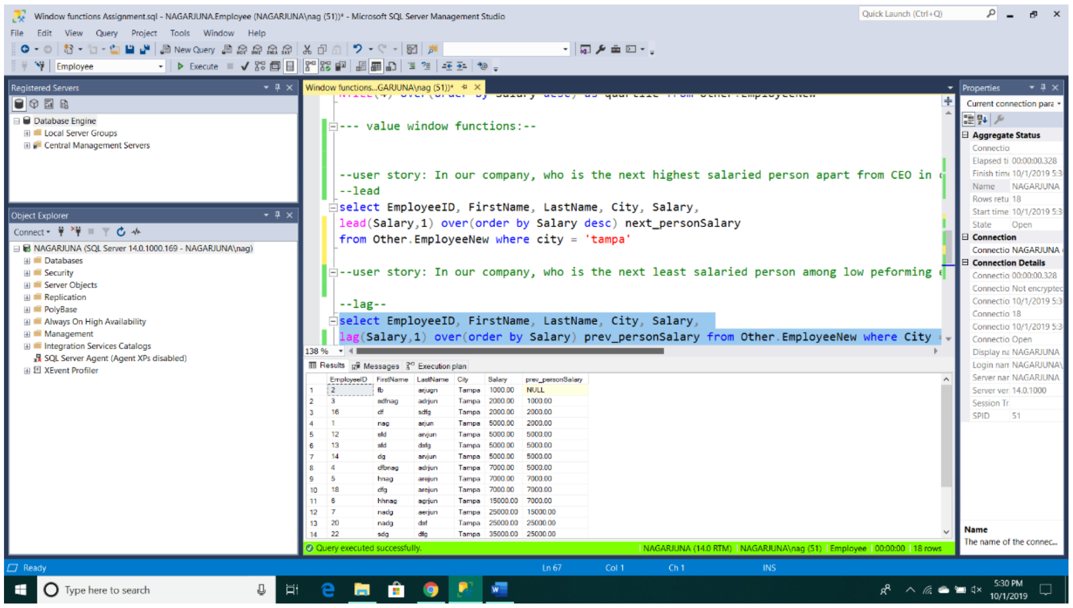
```
--user story: In our company, who is the next highest salaried person apart from CEO in
--lead
select EmployeeID, FirstName, LastName, City, Salary,
lead(Salary,1) over(order by Salary) next_personSalary
from Other.EmployeeNew where city = 'tampa'

--user story: In our company, who is the next least salaried person among low peforming

--lag--
select EmployeeID, FirstName, LastName, City, Salary,
lag(Salary,1) over(order by Salary) prev_personSalary from Other.EmployeeNew where City

--First_value and last_value
```

	EmployeeID	FirstName	LastName	City	Salary	next_personSalary
1	2	fb	arjugn	Tampa	1000.00	2000.00
2	3	sdfnag	adrjun	Tampa	2000.00	2000.00
3	16	df	sdfg	Tampa	2000.00	5000.00
4	1	nag	arjun	Tampa	5000.00	5000.00
5	12	sfd	arvjun	Tampa	5000.00	5000.00
6	13	sfd	dsfg	Tampa	5000.00	5000.00
7	14	dg	arvjun	Tampa	5000.00	7000.00
8	4	dfbnag	adrjun	Tampa	7000.00	7000.00
9	5	hnag	arejun	Tampa	7000.00	7000.00
10	18	dfg	arejun	Tampa	7000.00	15000.00
11	6	hhnag	agrjun	Tampa	15000.00	25000.00
12	7	nadg	aerjun	Tampa	25000.00	25000.00
13	20	nadg	dsf	Tampa	25000.00	35000.00
14	22	sdg	dfg	Tampa	35000.00	35000.00

Query executed successfully.

In screenshot above, we can see the next highest salaried employee

User story: In our company, who is the next least salaried person apart from the worst performing employee in our city



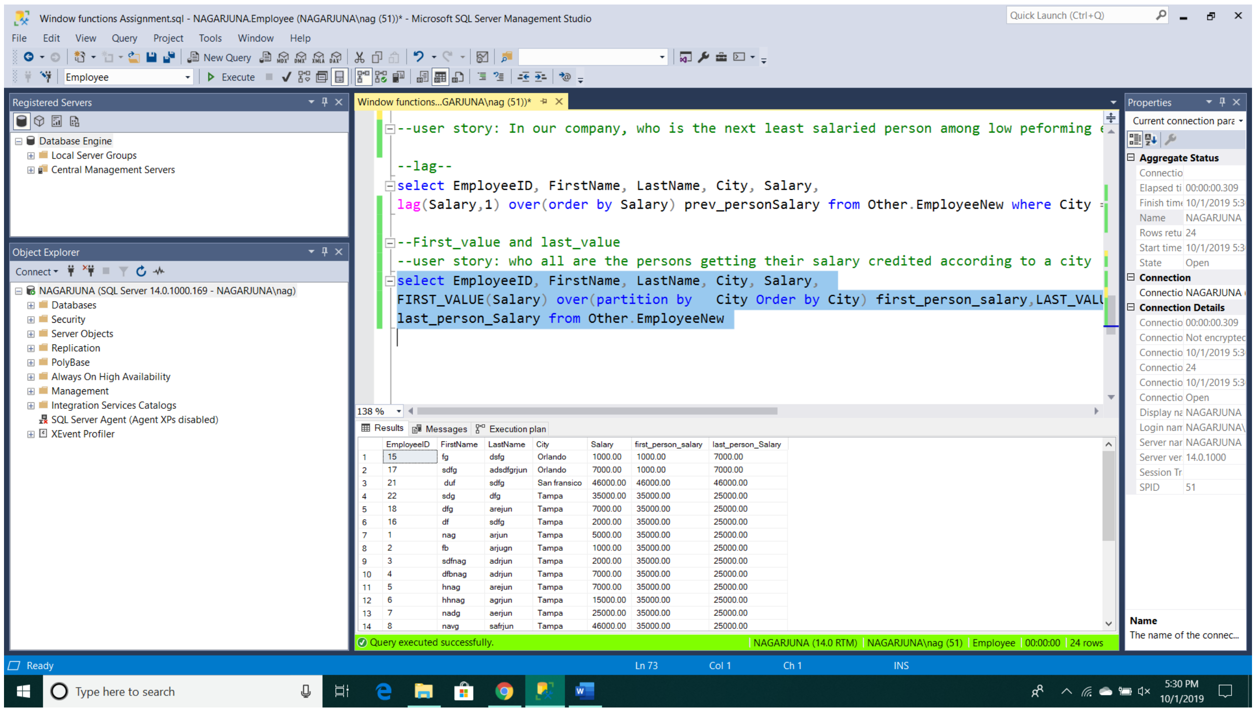
```
--- value window functions:--


--user story: In our company, who is the next highest salaried person apart from CEO in
--lead
select EmployeeID, FirstName, LastName, City, Salary,
lead(Salary,1) over(order by Salary desc) next_personSalary
from Other.EmployeeNew where city = 'tampa'

--user story: In our company, who is the next least salaried person among low peforming

--lag--
select EmployeeID, FirstName, LastName, City, Salary,
lag(Salary,1) over(order by Salary) prev_personSalary from Other.EmployeeNew where City
```

	EmployeeID	FirstName	LastName	City	Salary	prev_personSalary
1	2	fb	arjugn	Tampa	1000.00	NULL
2	3	sdfnag	adrjun	Tampa	2000.00	1000.00
3	16	df	sdfg	Tampa	2000.00	2000.00
4	1	nag	arjun	Tampa	5000.00	2000.00
5	12	sfd	arvjun	Tampa	5000.00	5000.00
6	13	sfd	dsfg	Tampa	5000.00	5000.00
7	14	dg	arvjun	Tampa	5000.00	5000.00
8	4	dfbnag	adrjun	Tampa	7000.00	5000.00
9	5	hnag	arejun	Tampa	7000.00	7000.00
10	18	dfg	arejun	Tampa	7000.00	7000.00
11	6	hhnag	agrjun	Tampa	15000.00	7000.00
12	7	nadg	aerjun	Tampa	25000.00	15000.00
13	20	nadg	dsf	Tampa	25000.00	25000.00
14	22	sdg	dfg	Tampa	35000.00	25000.00

Query executed successfully.

Here, we can see the next least salaried employee.

User story: In our company, who is the first and last person to get their salary credited according to our payroll system



```
--user story: In our company, who is the next least salaried person among low peforming e

--lag--
select EmployeeID, FirstName, LastName, City, Salary,
lag(Salary,1) over(order by Salary) prev_personSalary from Other.EmployeeNew where City =

--First_value and last_value
--user story: who all are the persons getting their salary credited according to a city
select EmployeeID, FirstName, LastName, City, Salary,
FIRST_VALUE(Salary) over(partition by   City Order by City) first_person_salary,LAST_VAL
last_person_Salary from Other.EmployeeNew
```

	EmployeeID	FirstName	LastName	City	Salary	first_person_salary	last_person_Salary
1	15	fg	dsfg	Orlando	1000.00	1000.00	7000.00
2	17	sdfg	adsdfgrjun	Orlando	7000.00	1000.00	7000.00
3	21	duf	sdfg	San fransico	46000.00	46000.00	46000.00
4	22	sdg	dfg	Tampa	35000.00	35000.00	25000.00
5	18	dfg	arejun	Tampa	7000.00	35000.00	25000.00
6	16	df	sdfg	Tampa	2000.00	35000.00	25000.00
7	1	nag	arjun	Tampa	5000.00	35000.00	25000.00
8	2	fb	arjugn	Tampa	1000.00	35000.00	25000.00
9	3	sdfnag	adrjun	Tampa	2000.00	35000.00	25000.00
10	4	dfbnag	adrjun	Tampa	7000.00	35000.00	25000.00
11	5	hnag	arejun	Tampa	7000.00	35000.00	25000.00
12	6	hhnag	agrjun	Tampa	15000.00	35000.00	25000.00
13	7	nadg	aerjun	Tampa	25000.00	35000.00	25000.00
14	8	navg	safrjun	Tampa	46000.00	35000.00	25000.00

Query executed successfully.

Here, we can see the first and last employees in a city with highest and lowest salaries in that particular city.