AWWF

1. So we have our employee table…lets use that for now
2.
```
select *

from employee;
```

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob
1	1	Ankit	100	10000	4	39	1983-12-02
2	2	Mohit	100	15000	5	48	1974-12-02
3	3	Vikas	100	10000	4	37	1985-12-02
4	4	Rohit	100	5000	2	16	2008-12-02
5	5	Mudit	200	12000	6	55	1967-12-02
6	6	Agam	200	12000	2	14	2008-12-02
7	7	Sanjay	200	9000	2	13	2009-12-02
8	8	Ashish	200	5000	2	12	2010-12-02
9	9	Mukesh	300	6000	6	51	1971-12-02
10	10	Rakesh	700	7000	6	50	1972-12-02
11	11	Ramesh	300	8000	6	52	1970-12-02

3. To find the employees avg salaries on each department ..previously we have used aggregate function and sub query and joins to achieve this
4. But we can achieve this using window function without usng any subquery
5. While using aggregate function and window analytic function…there is no need to give order by
6. But with non aggregate functions like row_number(),rank(),dense_rank() etc…we have to give order by
7. Check below query..for avg(salary)
8.
```
select *
,avg(salary) over(partition by dept_id) as avg_salary
from employee;
```

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	avg_salary
1	1	Ankit	100	10000	4	39	1983-12-02	10000
2	2	Mohit	100	15000	5	48	1974-12-02	10000
3	3	Vikas	100	10000	4	37	1985-12-02	10000
4	4	Rohit	100	5000	2	16	2008-12-02	10000
5	5	Mudit	200	12000	6	55	1967-12-02	9500
6	6	Agam	200	12000	2	14	2008-12-02	9500
7	7	Sanjay	200	9000	2	13	2009-12-02	9500
8	8	Ashish	200	5000	2	12	2010-12-02	9500
9	9	Mukesh	300	6000	6	51	1971-12-02	7000
10	11	Ramesh	300	8000	6	52	1970-12-02	7000
11	10	Rakesh	700	7000	6	50	1972-12-02	7000

9. Used max aggregate function

```
select *
,avg(salary) over(partition by dept_id) as avg_salary
,max(salary) over(partition by dept_id) as max_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	avg_salary	max_salary
1	1	Ankit	100	10000	4	39	1983-12-02	10000	15000
2	2	Mohit	100	15000	5	48	1974-12-02	10000	15000
3	3	Vikas	100	10000	4	37	1985-12-02	10000	15000
4	4	Rohit	100	5000	2	16	2006-12-02	10000	15000
5	5	Mudit	200	12000	6	55	1967-12-02	9500	12000
6	6	Agam	200	12000	2	14	2008-12-02	9500	12000
7	7	Sanjay	200	9000	2	13	2009-12-02	9500	12000
8	8	Ashish	200	5000	2	12	2010-12-02	9500	12000
9	9	Mukesh	300	6000	6	51	1971-12-02	7000	8000
10	11	Ramesh	300	8000	6	52	1970-12-02	7000	8000
11	10	Rakesh	700	7000	6	50	1972-12-02	7000	7000

10.

11. Used count function

```
select *
,avg(salary) over(partition by dept_id) as avg_salary
,count(salary) over(partition by dept_id) as max_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	avg_salary	max_salary
1	1	Ankit	100	10000	4	39	1983-12-02	10000	4
2	2	Mohit	100	15000	5	48	1974-12-02	10000	4
3	3	Vikas	100	10000	4	37	1985-12-02	10000	4
4	4	Rohit	100	5000	2	16	2006-12-02	10000	4
5	5	Mudit	200	12000	6	55	1967-12-02	9500	4
6	6	Agam	200	12000	2	14	2008-12-02	9500	4
7	7	Sanjay	200	9000	2	13	2009-12-02	9500	4
8	8	Ashish	200	5000	2	12	2010-12-02	9500	4
9	9	Mukesh	300	6000	6	51	1971-12-02	7000	2
10	11	Ramesh	300	8000	6	52	1970-12-02	7000	2
11	10	Rakesh	700	7000	6	50	1972-12-02	7000	1

12.

13. Using sum function without order by

14.
```
select *
,sum(salary) over(partition by dept_id ) as avg_salary
--,sum(salary) over(partition by dept_id order by emp_age asc) as avg_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	avg_salary
1	1	Ankit	100	10000	4	39	1983-12-02	40000
2	2	Mohit	100	15000	5	48	1974-12-02	40000
3	3	Vikas	100	10000	4	37	1985-12-02	40000
4	4	Rohit	100	5000	2	16	2006-12-02	40000
5	5	Mudit	200	12000	6	55	1967-12-02	38000
6	6	Agam	200	12000	2	14	2008-12-02	38000
7	7	Sanjay	200	9000	2	13	2009-12-02	38000
8	8	Ashish	200	5000	2	12	2010-12-02	38000
9	9	Mukesh	300	6000	6	51	1971-12-02	14000
10	11	Ramesh	300	8000	6	52	1970-12-02	14000
11	10	Rakesh	700	7000	6	50	1972-12-02	7000

15. Using sum(salary) with order by
16.
```
select *
,sum(salary) over(partition by dept_id ) as sum_salary
,sum(salary) over(partition by dept_id order by emp_age asc) as salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	sum_salary	salary
1	4	Rohit	100	5000	2	16	2006-12-02	40000	5000
2	3	Vikas	100	10000	4	37	1985-12-02	40000	15000
3	1	Ankit	100	10000	4	39	1983-12-02	40000	25000
4	2	Mohit	100	15000	5	48	1974-12-02	40000	40000
5	8	Ashish	200	5000	2	12	2010-12-02	38000	5000
6	7	Sanjay	200	9000	2	13	2009-12-02	38000	14000
7	6	Agam	200	12000	2	14	2008-12-02	38000	26000
8	5	Mudit	200	12000	6	55	1967-12-02	38000	38000
9	9	Mukesh	300	6000	6	51	1971-12-02	14000	6000
10	11	Ramesh	300	8000	6	52	1970-12-02	14000	14000
11	10	Rakesh	700	7000	6	50	1972-12-02	7000	7000

17. Here in the above query..we used order by emp_age…so for each salary corresponding to age..its adds up with next's age salary in that dept window ..see query output and understand
18. Used sum(salary) and order by emp_id ….here now it gives running sum based on emp_id..in that dept_id window

19.
```
select *
,sum(salary) over(partition by dept_id ) as sum_salary
,sum(salary) over(partition by dept_id order by emp_id asc) as salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	sum_salary	salary
1	1	Ankit	100	10000	4	39	1983-12-02	40000	10000
2	2	Mohit	100	15000	5	48	1974-12-02	40000	25000
3	3	Vikas	100	10000	4	37	1985-12-02	40000	35000
4	4	Rohit	100	5000	2	16	2006-12-02	40000	40000
5	5	Mudil	200	12000	6	55	1967-12-02	38000	12000
6	6	Agam	200	12000	2	14	2008-12-02	38000	24000
7	7	Sanjay	200	9000	2	13	2009-12-02	38000	33000
8	8	Ashish	200	5000	2	12	2010-12-02	38000	38000
9	9	Mukesh	300	6000	6	51	1971-12-02	14000	6000
10	11	Ramesh	300	8000	6	52	1970-12-02	14000	14000
11	10	Rakesh	700	7000	6	50	1972-12-02	7000	7000

20. Using aggregate function without partition
21.
```
select *
,sum(salary) over(partition by dept_id ) as sum_salary
,sum(salary) over(partition by dept_id order by emp_id asc) as dep_running_salary
,sum(salary) over(order by emp_id asc) as dep_running_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	sum_salary	dep_running_salary	dep_running_salary
1	1	Ankit	100	10000	4	39	1983-12-02	40000	10000	10000
2	2	Mohit	100	15000	5	48	1974-12-02	40000	25000	25000
3	3	Vikas	100	10000	4	37	1985-12-02	40000	35000	35000
4	4	Rohit	100	5000	2	16	2006-12-02	40000	40000	40000
5	5	Mudil	200	12000	6	55	1967-12-02	38000	12000	52000
6	6	Agam	200	12000	2	14	2008-12-02	38000	24000	64000
7	7	Sanjay	200	9000	2	13	2009-12-02	38000	33000	73000
8	8	Ashish	200	5000	2	12	2010-12-02	38000	38000	78000
9	9	Mukesh	300	6000	6	51	1971-12-02	14000	6000	84000
10	10	Rakesh	700	7000	6	50	1972-12-02	7000	7000	91000
11	11	Ramesh	300	8000	6	52	1970-12-02	14000	14000	99000

22. Using 1st..max(salary) with partition..without order by as sum_salary…2nd..with both partition and order by…3rd..without partition

```
select *
,max(salary) over(partition by dept_id ) as sum_salary
,max(salary) over(partition by dept_id order by emp_id asc) as dep_running_salary
,max(salary) over(order by emp_id asc) as dep_running_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	sum_salary	dep_running_salary	dep_running_salary
1	1	Ankit	100	10000	4	39	1983-12-02	15000	10000	10000
2	2	Mohit	100	15000	5	48	1974-12-02	15000	15000	15000
3	3	Vikas	100	10000	4	37	1985-12-02	15000	15000	15000
4	4	Rohit	100	5000	2	16	2006-12-02	15000	15000	15000
5	5	Mudit	200	12000	6	55	1967-12-02	12000	12000	15000
6	6	Agam	200	12000	2	14	2008-12-02	12000	12000	15000
7	7	Sanjay	200	9000	2	13	2009-12-02	12000	12000	15000
8	8	Ashish	200	5000	2	12	2010-12-02	12000	12000	15000
9	9	Mukesh	300	6000	6	51	1971-12-02	8000	8000	15000
10	10	Rakesh	700	7000	6	50	1972-12-02	7000	7000	15000
11	11	Ramesh	300	8000	6	52	1970-12-02	8000	8000	15000

23.

24. max(salary) over(order by salary asc) max_running_salary

```
select *
,max(salary) over(partition by dept_id ) as sum_salary
,max(salary) over(order by salary asc) as max_running_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	sum_salary	max_running_salary
1	4	Rohit	100	5000	2	16	2006-12-02	15000	5000
2	8	Ashish	200	5000	2	12	2010-12-02	12000	5000
3	9	Mukesh	300	6000	6	51	1971-12-02	8000	6000
4	10	Rakesh	700	7000	6	50	1972-12-02	7000	7000
5	11	Ramesh	300	8000	6	52	1970-12-02	8000	8000
6	7	Sanjay	200	9000	2	13	2009-12-02	12000	9000
7	3	Vikas	100	10000	4	37	1985-12-02	15000	10000
8	1	Ankit	100	10000	4	39	1983-12-02	15000	10000
9	5	Mudit	200	12000	6	55	1967-12-02	12000	12000
10	6	Agam	200	12000	2	14	2008-12-02	12000	12000
11	2	Mohit	100	15000	5	48	1974-12-02	15000	15000

25.

26. Here in max_running_salary…as we are going in asc..so we get highest salary for each row..check output and understand

27. max(salary) over(order by salary dsc) max_running_salary…now as 15000 is the highest salary…it gives us all 15000's as max_running_salary

```
select *
,max(salary) over(partition by dept_id ) as sum_salary
,max(salary) over(order by salary desc) as max_running_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	sum_salary	max_running_salary
1	2	Mohit	100	15000	5	48	1974-12-02	15000	15000
2	5	Mudit	200	12000	6	55	1967-12-02	12000	15000
3	6	Agam	200	12000	2	14	2008-12-02	12000	15000
4	3	Vikas	100	10000	4	37	1985-12-02	15000	15000
5	1	Ankit	100	10000	4	39	1983-12-02	15000	15000
6	7	Sanjay	200	9000	2	13	2009-12-02	12000	15000
7	11	Ramesh	300	8000	6	52	1970-12-02	8000	15000
8	10	Rakesh	700	7000	6	50	1972-12-02	7000	15000
9	9	Mukesh	300	6000	6	51	1971-12-02	8000	15000
10	4	Rohit	100	5000	2	16	2006-12-02	15000	15000
11	8	Ashish	200	5000	2	12	2010-12-02	12000	15000

28.
29. sum(salary) ..over order by salary DESC..

```
select *
,sum(salary) over(order by salary DESC) as running_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	running_salary
1	2	Mohit	100	15000	5	48	1974-12-02	15000
2	5	Mudit	200	12000	6	55	1967-12-02	39000
3	6	Agam	200	12000	2	14	2008-12-02	39000
4	3	Vikas	100	10000	4	37	1985-12-02	59000
5	1	Ankit	100	10000	4	39	1983-12-02	59000
6	7	Sanjay	200	9000	2	13	2009-12-02	68000
7	11	Ramesh	300	8000	6	52	1970-12-02	76000
8	10	Rakesh	700	7000	6	50	1972-12-02	83000
9	9	Mukesh	300	6000	6	51	1971-12-02	89000
10	4	Rohit	100	5000	2	16	2006-12-02	99000
11	8	Ashish	200	5000	2	12	2010-12-02	99000

30.
31. Here if we see…first row of running salary has 15000..perfectly fine…but in 2nd we have 39000 ..because…if there's any consecutive duplicate values in order by col…it sums up together at once..see pic and understand
32. Similarly for order by salary ASC

33.
```
select *
,sum(salary) over(order by salary) as running_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	running_salary
1	4	Rohit	100	5000	2	16	2006-12-02	10000
2	8	Ashish	200	5000	2	12	2010-12-02	10000
3	9	Mukesh	300	6000	6	51	1971-12-02	16000
4	10	Rakesh	700	7000	6	50	1972-12-02	23000
5	11	Ramesh	300	8000	6	52	1970-12-02	31000
6	7	Sanjay	200	9000	2	13	2009-12-02	40000
7	3	Vikas	100	10000	4	37	1985-12-02	50000
8	1	Ankit	100	10000	4	39	1983-12-02	60000
9	5	Mudit	200	12000	6	55	1967-12-02	84000
10	6	Agam	200	12000	2	14	2008-12-02	84000
11	2	Mohit	100	15000	5	48	1974-12-02	99000

34. If we want to tackle this problem…we shud give 2 col's in order_by
35.
```
select *
,sum(salary) over(order by salary,emp_id DESC) as running_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	running_salary
1	8	Ashish	200	5000	2	12	2010-12-02	5000
2	4	Rohit	100	5000	2	16	2006-12-02	10000
3	9	Mukesh	300	6000	6	51	1971-12-02	16000
4	10	Rakesh	700	7000	6	50	1972-12-02	23000
5	11	Ramesh	300	8000	6	52	1970-12-02	31000
6	7	Sanjay	200	9000	2	13	2009-12-02	40000
7	3	Vikas	100	10000	4	37	1985-12-02	50000
8	1	Ankit	100	10000	4	39	1983-12-02	60000
9	6	Agam	200	12000	2	14	2008-12-02	72000
10	5	Mudit	200	12000	6	55	1967-12-02	84000
11	2	Mohit	100	15000	5	48	1974-12-02	99000

Query executed successfully.

36. In the above query…we used 2 col's in order by…first is order by salary asc..and 2nd is emp_id DESC
37. Here we have calculated avg running salary..practice query and understand

```
select *
,avg(salary) over(partition by dept_id order by emp_id ) as running_salary
from employee;

select * from employee
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	running_salary
1	1	Ankit	100	10000	4	39	1983-12-02	10000
2	2	Mohit	100	15000	5	48	1974-12-02	12500
3	3	Vikas	100	10000	4	37	1985-12-02	11666
4	4	Rohit	100	5000	2	16	2006-12-02	10000
5	5	Mudit	200	12000	6	55	1967-12-02	12000
6	6	Agam	200	12000	2	14	2008-12-02	12000
7	7	Sanjay	200	9000	2	13	2009-12-02	11000
8	8	Ashish	200	5000	2	12	2010-12-02	9500
9	9	Mukesh	300	6000	6	51	1971-12-02	6000
10	11	Ramesh	300	8000	6	52	1970-12-02	7000
11	10	Rakesh	700	7000	6	50	1972-12-02	7000

38.

39. Running count

```
select *
,count(salary) over(partition by dept_id order by emp_id ) as running_salary
from employee;

select * from employee
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	running_salary
1	1	Ankit	100	10000	4	39	1983-12-02	1
2	2	Mohit	100	15000	5	48	1974-12-02	2
3	3	Vikas	100	10000	4	37	1985-12-02	3
4	4	Rohit	100	5000	2	16	2006-12-02	4
5	5	Mudit	200	12000	6	55	1967-12-02	1
6	6	Agam	200	12000	2	14	2008-12-02	2
7	7	Sanjay	200	9000	2	13	2009-12-02	3
8	8	Ashish	200	5000	2	12	2010-12-02	4
9	9	Mukesh	300	6000	6	51	1971-12-02	1
10	11	Ramesh	300	8000	6	52	1970-12-02	2
11	10	Rakesh	700	7000	6	50	1972-12-02	1

40.

41. If we remove order by from these queries..then there will be no running values
42. We can check the diff here

43.

```
select *
,sum(salary) over(partition by dept_id ) as dep_salary
,sum(salary) over(partition by dept_id order by emp_id) as ep_running_salary
from employee;

select * from employee
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	dep_salary	ep_running_salary
1	1	Ankit	100	10000	4	39	1983-12-02	40000	10000
2	2	Mohit	100	15000	5	48	1974-12-02	40000	25000
3	3	Vikas	100	10000	4	37	1985-12-02	40000	35000
4	4	Rohit	100	5000	2	16	2006-12-02	40000	40000
5	5	Mudit	200	12000	6	55	1967-12-02	38000	12000
6	6	Agam	200	12000	2	14	2008-12-02	38000	24000
7	7	Sanjay	200	9000	2	13	2009-12-02	38000	33000
8	8	Ashish	200	5000	2	12	2010-12-02	38000	38000
9	9	Mukesh	300	6000	6	51	1971-12-02	14000	6000
10	11	Ramesh	300	8000	6	52	1970-12-02	14000	14000
11	10	Rakesh	700	7000	6	50	1972-12-02	7000	7000

44. In the below query we are calculating sum of current row and previous 2 row's sum
45. Preceding and current row
46.

```
select *
,sum(salary) over( order by emp_id rows between 2 preceding and current row ) as rolling_3_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	running_salary
1	1	Ankit	100	10000	4	39	1983-12-02	10000
2	2	Mohit	100	15000	5	48	1974-12-02	25000
3	3	Vikas	100	10000	4	37	1985-12-02	35000
4	4	Rohit	100	5000	2	16	2006-12-02	30000
5	5	Mudit	200	12000	6	55	1967-12-02	27000
6	6	Agam	200	12000	2	14	2008-12-02	29000
7	7	Sanjay	200	9000	2	13	2009-12-02	33000
8	8	Ashish	200	5000	2	12	2010-12-02	26000
9	9	Mukesh	300	6000	6	51	1971-12-02	20000
10	10	Rakesh	700	7000	6	50	1972-12-02	18000
11	11	Ramesh	300	8000	6	52	1970-12-02	21000

Query executed successfully.

47. Next we calculated sum of previous row,current row,and next row…using 1 preceding and 1 following..see pic

48.
```
select *
,sum(salary) over( order by emp_id rows between 1 preceding and 1 following ) as rolling_3_salary
from employee;
```

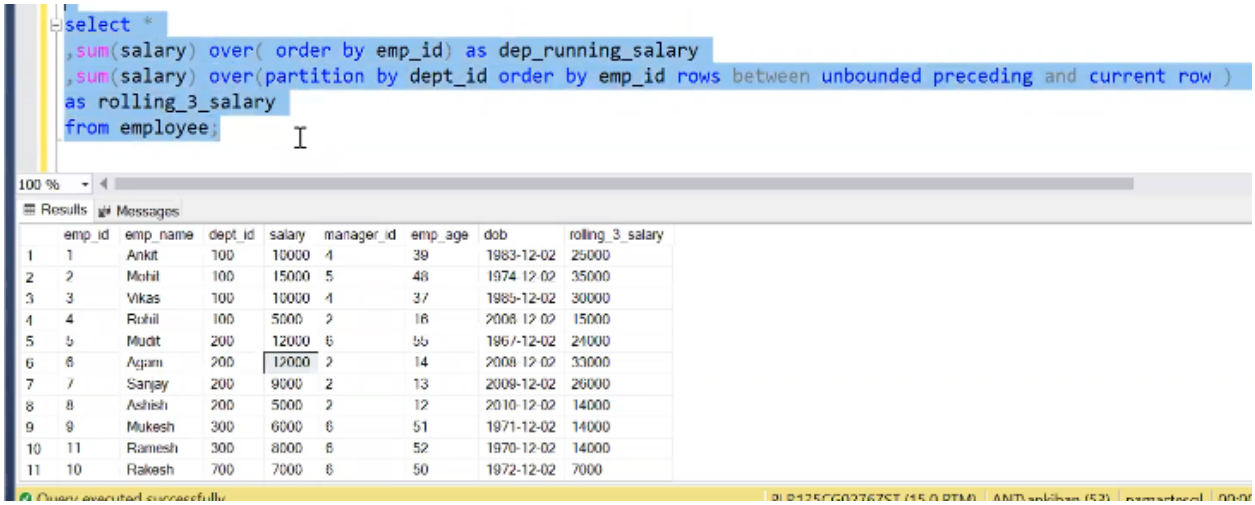
	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	rolling_3_salary
1	1	Ankit	100	10000	4	39	1983-12-02	25000
2	2	Mohit	100	15000	5	48	1974-12-02	35000
3	3	Vikas	100	10000	4	37	1985-12-02	30000
4	4	Rohit	100	5000	2	16	2006-12-02	27000
5	5	Mudit	200	12000	6	55	1967-12-02	29000
6	6	Agam	200	12000	2	14	2008-12-02	33000
7	7	Sanjay	200	9000	2	13	2009-12-02	26000
8	8	Ashish	200	5000	2	12	2010-12-02	20000
9	9	Mukesh	300	6000	6	51	1971-12-02	18000
10	10	Rakesh	700	7000	6	50	1972-12-02	21000
11	11	Ramesh	300	8000	6	52	1970-12-02	15000

49. Sum of each row value = prev_row(salary) + current_row(salary) + next_row(salary) ..for the above output
50. Rows between 5 following and 10 following
51.
```
select *
,sum(salary) over( order by emp_id rows between 5 following and 10 following ) as rolling_3_salary
from employee;
```

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	rolling_3_salary
1	1	Ankit	100	10000	4	39	1983-12-02	47000
2	2	Mohit	100	15000	5	48	1974-12-02	35000
3	3	Vikas	100	10000	4	37	1985-12-02	26000
4	4	Rohit	100	5000	2	16	2006-12-02	21000
5	5	Mudit	200	12000	6	55	1967-12-02	15000
6	6	Agam	200	12000	2	14	2008-12-02	8000
7	7	Sanjay	200	9000	2	13	2009-12-02	NULL
8	8	Ashish	200	5000	2	12	2010-12-02	NULL
9	9	Mukesh	300	6000	6	51	1971-12-02	NULL
10	10	Rakesh	700	7000	6	50	1972-12-02	NULL
11	11	Ramesh	300	8000	6	52	1970-12-02	NULL

52. If we use partition on above query ..we get all nulls in rolling_3_salary col
53.
```
select *
,sum(salary) over(partition by dept_id order by emp_id rows between 5 following and 10 following ) as rolling_3_s
from employee;
```

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	rolling_3_salary
1	1	Ankit	100	10000	4	39	1983-12-02	NULL
2	2	Mohit	100	15000	5	48	1974-12-02	NULL
3	3	Vikas	100	10000	4	37	1985-12-02	NULL
4	4	Rohit	100	5000	2	16	2006-12-02	NULL
5	5	Mudit	200	12000	6	55	1967-12-02	NULL
6	6	Agam	200	12000	2	14	2008-12-02	NULL
7	7	Sanjay	200	9000	2	13	2009-12-02	NULL
8	8	Ashish	200	5000	2	12	2010-12-02	NULL
9	9	Mukesh	300	6000	6	51	1971-12-02	NULL
10	11	Ramesh	300	8000	6	52	1970-12-02	NULL
11	10	Rakesh	700	7000	6	50	1972-12-02	NULL

54. Unbounded preceding and current row
55. 

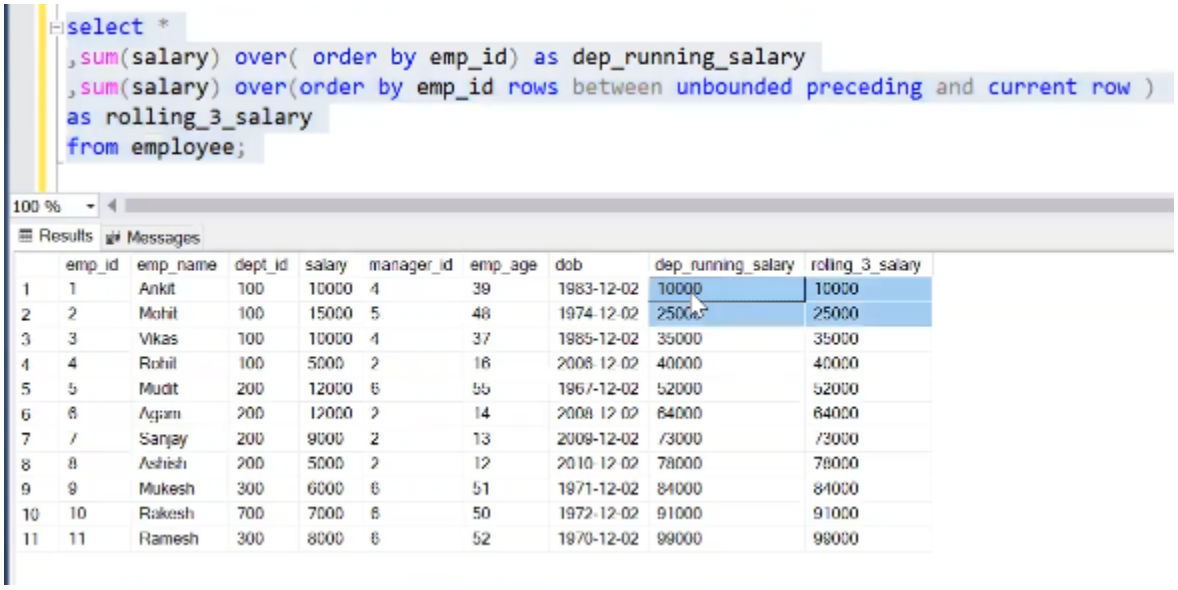
```
select *
,sum(salary) over( order by emp_id) as dep_running_salary
,sum(salary) over(partition by dept_id order by emp_id rows between unbounded preceding and current row )
as rolling_3_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	rolling_3_salary
1	1	Ankit	100	10000	4	39	1983-12-02	25000
2	2	Mohit	100	15000	5	48	1974-12-02	35000
3	3	Vikas	100	10000	4	37	1985-12-02	30000
4	4	Rohit	100	5000	2	16	2008-12-02	15000
5	5	Mudit	200	12000	6	55	1967-12-02	24000
6	6	Agam	200	12000	2	14	2008-12-02	33000
7	7	Sanjay	200	9000	2	13	2009-12-02	26000
8	8	Ashish	200	5000	2	12	2010-12-02	14000
9	9	Mukesh	300	6000	6	51	1971-12-02	14000
10	11	Ramesh	300	8000	6	52	1970-12-02	14000
11	10	Rakesh	700	7000	6	50	1972-12-02	7000

Query executed successfully.

56. Here we have sum up all the previous rows with current row…similar to dep_running_salary col's query…see pic and understand
57. 

```
select *
,sum(salary) over( order by emp_id) as dep_running_salary
,sum(salary) over(order by emp_id rows between unbounded preceding and current row )
as rolling_3_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	dep_running_salary	rolling_3_salary
1	1	Ankit	100	10000	4	39	1983-12-02	10000	10000
2	2	Mohit	100	15000	5	48	1974-12-02	25000	25000
3	3	Vikas	100	10000	4	37	1985-12-02	35000	35000
4	4	Rohit	100	5000	2	16	2008-12-02	40000	40000
5	5	Mudit	200	12000	6	55	1967-12-02	52000	52000
6	6	Agam	200	12000	2	14	2008-12-02	64000	64000
7	7	Sanjay	200	9000	2	13	2009-12-02	73000	73000
8	8	Ashish	200	5000	2	12	2010-12-02	78000	78000
9	9	Mukesh	300	6000	6	51	1971-12-02	84000	84000
10	10	Rakesh	700	7000	6	50	1972-12-02	91000	91000
11	11	Ramesh	300	8000	6	52	1970-12-02	99000	99000

58. See above pic and understand
59. Unfounded preceding and unbounded following…it gives us sum of everything…all prev rows and all next rows and current row

```
select *
,sum(salary) over( order by emp_id) as dep_running_salary
,sum(salary) over(order by emp_id rows between unbounded preceding and unbounded following )
as rolling_3_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	dep_running_salary	rolling_3_salary
1	1	Ankit	100	10000	4	39	1983-12-02	10000	99000
2	2	Mohit	100	15000	5	48	1974-12-02	25000	99000
3	3	Vikas	100	10000	4	37	1985-12-02	35000	99000
4	4	Rohit	100	5000	2	16	2006-12-02	40000	99000
5	5	Mudit	200	12000	6	55	1967-12-02	52000	99000
6	6	Agam	200	12000	2	14	2008-12-02	64000	99000
7	7	Sanjay	200	9000	2	13	2009-12-02	73000	99000
8	8	Ashish	200	5000	2	12	2010-12-02	78000	99000
9	9	Mukesh	300	6000	6	51	1971-12-02	84000	99000
10	10	Rakesh	700	7000	6	50	1972-12-02	91000	99000
11	11	Ramesh	300	8000	6	52	1970-12-02	99000	99000

60.

61. Unbounded preceding and unbounded following with partition by dept_id

```
select *
,sum(salary) over( order by emp_id) as dep_running_salary
,sum(salary) over(partition by dept_id order by emp_id rows between unbounded preceding and unbounded following )
as total_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	dep_running_salary	total_salary
1	1	Ankit	100	10000	4	39	1983-12-02	10000	40000
2	2	Mohit	100	15000	5	48	1974-12-02	25000	40000
3	3	Vikas	100	10000	4	37	1985-12-02	35000	40000
4	4	Rohit	100	5000	2	16	2006-12-02	40000	40000
5	5	Mudit	200	12000	6	55	1967-12-02	52000	38000
6	6	Agam	200	12000	2	14	2008-12-02	64000	38000
7	7	Sanjay	200	9000	2	13	2009-12-02	73000	38000
8	8	Ashish	200	5000	2	12	2010-12-02	78000	38000
9	9	Mukesh	300	6000	6	51	1971-12-02	84000	14000
10	11	Ramesh	300	8000	6	52	1970-12-02	99000	14000
11	10	Rakesh	700	7000	6	50	1972-12-02	91000	7000

62.

63. Next we have first value and last value
64. First value gives …first value in the corresponding column(salary)
65. Coming to last value…as it searches from top to bottom…it thinks..it every value coming from top to bottom is last value…see pic and understand

```
select *
,first_value(salary) over(order by salary) as first_salary
,last_value(salary) over(order by salary) as last_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	first_salary	last_salary
1	4	Rohit	100	5000	2	16	2006-12-02	5000	5000
2	8	Ashish	200	5000	2	12	2010-12-02	5000	5000
3	9	Mukesh	300	6000	6	51	1971-12-02	5000	6000
4	10	Rakesh	700	7000	8	50	1972-12-02	5000	7000
5	11	Ramesh	300	8000	6	52	1970-12-02	5000	8000
6	7	Sanjay	200	9000	2	13	2009-12-02	5000	9000
7	3	Vikas	100	10000	4	37	1986-12-02	5000	10000
8	1	Ankit	100	10000	4	39	1983-12-02	5000	10000
9	5	Mudit	200	12000	6	55	1967-12-02	5000	12000
10	6	Agam	200	12000	2	14	2008-12-02	5000	12000
11	2	Mohit	100	15000	5	48	1974-12-02	5000	15000

66.

67. If we give unbounded access to last value ….then it has access to the entire col of values…and it finds the last value…and gives us that value in last_salary col

```
select *
,first_value(salary) over(order by salary) as first_salary
,last_value(salary) over(order by salary rows between unbounded preceding and unbounded following ) as last_salary
from employee;
```

100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	first_salary	last_salary
1	4	Rohit	100	5000	2	16	2006-12-02	5000	15000
2	8	Ashish	200	5000	2	12	2010-12-02	5000	15000
3	9	Mukesh	300	6000	6	51	1971-12-02	5000	15000
4	10	Rakesh	700	7000	8	50	1972-12-02	5000	15000
5	11	Ramesh	300	8000	6	52	1970-12-02	5000	15000
6	7	Sanjay	200	9000	2	13	2009-12-02	5000	15000
7	3	Vikas	100	10000	4	37	1986-12-02	5000	15000
8	1	Ankit	100	10000	4	39	1983-12-02	5000	15000
9	5	Mudit	200	12000	6	55	1967-12-02	5000	15000
10	6	Agam	200	12000	2	14	2008-12-02	5000	15000
11	2	Mohit	100	15000	5	48	1974-12-02	5000	15000

68.

69. We can also get last_salary….by

```
,first_value(salary) over(order by salary desc) as last_salary
```

70. Next we are retrieving order_id,sales,sum(sales) using over(order by order_id) as running_sales

71.
```
select order_id,sales,sum(sales) over(order by order_id) as running_sales
from orders
```

	order_id	sales	running_sales
1	CA-2018-100006	377.97	377.97
2	CA-2018-100090	502.488	1077.162
3	CA-2018-100090	196.704	1077.162
4	CA-2018-100293	91.056	1168.218
5	CA-2018-100328	3.928	1172.146
6	CA-2018-100363	2.368	1193.522
7	CA-2018-100363	19.008	1193.522
8	CA-2018-100391	14.62	1208.142
9	CA-2018-100678	2.688	1905.216
10	CA-2018-100678	317.058	1905.216
11	CA-2018-100678	149.352	1905.216

72. Here we retrieved running sales for each order_id…
73. Running sales is nothing but…sum of sales of current row + sales of prev_rows
74. Here in the above query…it is not handling duplicates properly
75. To handles duplicates…we use unbounded preceding and current row..check below
76.
```
select order_id,sales,sum(sales) over(order by order_id rows between unbounded preceding and current row) as runni
from orders
```

	order_id	sales	running_sales
1	CA-2018-100006	377.97	377.97
2	CA-2018-100090	502.488	880.458
3	CA-2018-100090	196.704	1077.162
4	CA-2018-100293	91.056	1168.218
5	CA-2018-100328	3.928	1172.146
6	CA-2018-100363	2.368	1174.514
7	CA-2018-100363	19.008	1193.522
8	CA-2018-100391	14.62	1208.142
9	CA-2018-100678	2.688	1210.83
10	CA-2018-100678	317.058	1527.888
11	CA-2018-100678	149.352	1677.24

77. The above query..even if duplicates are there…it will handle them one by one
78. Try using group by on above query(task for my self)
79. Next we are getting total_sales by each month in that year
80.
```
select datepart(year,order_date) as year_order,datepart(month,order_date) as month_order,sum(sales) as total_sales
from orders
group by datepart(year,order_date),datepart(month,order_date)
order by year_order,month_order
```

	year_order	month_order	total_sales
1	2018	1	14236.895
2	2018	2	4519.892
3	2018	3	55691.009
4	2018	4	28295.345
5	2018	5	23648.287
6	2018	6	34595.1276
7	2018	7	33946.393
8	2018	8	27909.4685
9	2018	9	81777.3508
10	2018	10	31453.393
11	2018	11	78628.7166999999

81. Next from this result..we get rolling_3_sales..using the below query

```
with month_wise_sales as
(select datepart(year,order_date) as year_order,datepart(month,order_date) as month_order,sum(sales) as total_sales
from orders
group by datepart(year,order_date),datepart(month,order_date))
select
year_order,month_order,total_sales
,sum(total_sales) over(order by year_order,month_order rows between 2 preceding and current row) as rolling_3_sales
from month_wise_sales
```

100 %

Results | Messages

	year_order	month_order	total_sales	rolling_3_sales
1	2018	1	14236.895	14236.895
2	2018	2	4519.892	18756.787
3	2018	3	55681.009	74447.796
4	2018	4	28295.345	88506.246
5	2018	5	23648.287	107634.641
6	2018	6	34595.1276	86538.7596
7	2018	7	33946.393	92189.8076
8	2018	8	27909.4685	96450.9891
9	2018	9	81777.3508	143633.2123
10	2018	10	31453.393	141140.2123

Query executed successfully.

82.

83.