Windows Function.

- Windows function allows you to perform calculation ~~over~~ across a set of related rows (a "window") without grouping the result into a single row.

In simple terms:

It lets you calculate things like ranking, running totals, moving averages, or differences across rows.

- It is used with over() clause using partition by() & order by()

Example1 → Row-Number() → Rank Rows within group

```
Select *,
Row-number() over (partition by Employee order by amount
                   desc) as row num
from Sales.
```

???	marks	rownum
:	91	1
(	71	2
(	90	3
	89	4
1	89	5
1	89	6
.	95	7.

Rank → If there's tie next rank or ranks will be skipped.

```
select *,
rank() over(partition by subjects order by marks desc)
    as [marks rank]
from Students.
```

xyz	marks	Rank.
-	92	1
-	90	2
-	90	2
-	90	2
-	88	5
-	86	6
-	86	6
-	85	8

dense_rank → Next Ranks are not skipped in case if they are same.

```
Select *,
dense_Rank() over(order by marks Desc) [DR]
from students.
```

xyz	lmn	marks	DR
-	-	92	1
-	-	90	2
-	-	90	2
-	-	86	3
-	-	85	4
-	-	85	4
-	-	84	5.

running ~~moving~~ Sum/total of marks.

```
select *,
sum(marks) over(partition by subject order by marks) as
                [moving total]
from students.
```

moving Average marks

```
select *,                    partition by subject
avg(marks) over (order by marks desc) moving(Avg)
from students
```

Lead & Lag windows function.

To get the next months/row data/profit.

```
select *, monthNumber, monthname, sum(profit) as [Total profit]
lead (sum(profit)) over (order by monthnumber) as
        [next month's profit]
from profitdata.
group by monthnumber, monthname.
```

getting next month's chain salary using CTE

```
with Total profit as (
    select monthnumber, monthname,
    sum(profit) as Total Profit
    from sales
    group by monthnumber, monthname.
)
Select .
    monthnumber, month name, Total profit,
    lead (Total profit) over (order by monthnumber) as [Next.
                                    months profit],
    lead (Total profit) over (order by month number) — Total profit
                                    as chain profit.
from sales.
```

IsNull() & Coalesce function

Isnull()

```
select
email, phonenumber, address,
isnull(email, 'no email') as [refine email coln]
from customers
```

```
select
email,
isnull(phonenumber, 'No records found') as [phone-no]
from customers.
```

Strictly take only one argument.

Syntax

isnull (expression1, value)

2) coalesce.

```
select.
cust-id, firstname,
coalesce (email, phoneno, 'nocontact')
from customers
```

Syntax

coalesce (exp1, exp2, ... expn)

#firstvalue

When first_value is useful?

→ To find :→

→ First purchase of cust, first salary of employee. first login of user, first sale in region etc..

product launch price.

```
select
cust_id,
order_date,
first_value(order date) over(partition by cust_id.
                             order by order_date)
                             as [firstorder]
from orders.
```

Common table expression

- Common table expression is temporary results set that you can create references with a select, insert, update or delete statement.
- It is defined using with keyword & they make complex theory easy to write, understand & maintain by breaking them in simpler task.

```
with monthly Profit as (
select
cust-ID, monthnumber,
cust-name monthname.
sum(profit) as [Total Profit]
from profitdata
group by cust-ID, cust-name.
)
Select
monthNumber,
monthname,
Total profit,
lead(TotalProfit) over (order by monthNumber) As NextmonthProfit,
Lead(TotalProfit) over (order by monthNumber) - Total Profit
as profitchange.
from monthly profit
```

Factorial of number

```
with fact as (
 --anchor query
 select 1 as n.
union all
-- reccursive query
 select n+1 from fact where n <5.
 )
 select exp(sum(log(n1)) from [fact]
```

star pattern

```
 with stars as.(
  select 1 as n,
 cast('*' as varchar(max)) as pattern
 union all
 select n+1, patten + '*'
 from stars
 where n<5.
)
 select patten from stars.
```

~~empid → manager id hierarchy~~

Stored procedure.

~~with~~

↷ convert proc

```
Create procedure sp-test.
as
begin.
   select * from employees
end.
```

```
create proc procname
as
begin
      Body.
end.
```

sp-test.
Exec sp-test
execute sp-test

3 ways of recalling Stored procedure.