Window Function

**--\_\_\_\_\_\_\_ 4. Window function \_\_\_**

**Window function work with Over() -> order by**

1. **RANK()** - assigns a unique rank to each row based on the values in one or more columns. Rows with the same values get the same rank, and the next rank is skipped.

CREATE TABLE SalesData (

SalesPersonID INT,

SalesPerson VARCHAR(50),

SalesAmount DECIMAL(10, 2)

);

INSERT INTO SalesData (SalesPersonID, SalesPerson, SalesAmount)

VALUES

(1, 'John Smith', 5000.00),

(2, 'Jane Doe', 7500.00),

(3, 'Bob Johnson', 10000.00),

(4, 'Mary Wilson', 5000.00),

(5, 'Tom Jones', 2500.00),

(6, 'Sue Brown', 15000.00),

(7, 'Mike Davis', 12500.00),

(8, 'Jill Green', 8000.00),

(9, 'Bill Lee', 11000.00),

(10, 'Kate White', 9000.00);

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Clustured Index pr (Id) pr by Default Rank lgi hoti haa \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

select

\*,

Rank() over(order by salesPersonId) as 'rank'

from SalesData

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Rank Row base on Salery Amont \_\_\_\_\_\_\_\_\_\_\_\_\_

select

\*,

Rank() over (order by salesAmount desc) as 'rank'

from SalesData

1. **DENSE\_RANK()** –
   * + same as Rank.
     + But with Same Values Assign Same Rank Number

SELECT

DENSE\_RANK() OVER (ORDER BY SalesAmount DESC) AS 'Rank',

SalesPerson,

SalesAmount

FROM SalesData;

1. **NTILE()**
   * + **Divide the result into a specified number of group.**
     + **Assig the Group number (1,1,1,2,2,3,……)**

SELECT

NTILE(4) OVER (ORDER BY SalesAmount DESC) AS 'Quartile',

SalesPerson,

SalesAmount

FROM SalesData;

1. **PERCENT\_RANK()** - calculates the relative rank of each row within the result set as a percentage. The rank ranges from 0 to 1, with 0 being the lowest rank and 1 being the highest.

SELECT PERCENT\_RANK() OVER (ORDER BY SalesAmount DESC) AS 'PercentRank', SalesPerson, SalesAmount FROM SalesData;

1. **CUME\_DIST()** - calculates the cumulative distribution of a value within a set of values. The result is a percentage that represents the portion of the result set that is less than or equal to the current row.

SELECT CUME\_DIST() OVER (ORDER BY SalesAmount DESC) AS 'CumulativeDist', SalesPerson, SalesAmount FROM SalesData;

1. **LEAD()** - returns the value of a specified column for the next row in the result set. You can specify the number of rows ahead to look.

SELECT SalesPerson,

SalesAmount,

LEAD(SalesAmount, 1) OVER (ORDER BY SalesAmount DESC) AS 'NextSalesAmount'

FROM SalesData;

1. **LAG()** - returns the value of a specified column for the previous row in the result set. You can specify the number of rows behind to look.

SELECT

SalesPerson,

SalesAmount,

LAG(SalesAmount, 1) OVER (ORDER BY SalesAmount DESC) AS 'PrevSalesAmount'

FROM SalesData;

1. **ROW\_NUMBER()** - assigns a unique sequential number to each row within a result set.

Make the Rand base on Integer values

SELECT

ROW\_NUMBER() OVER (ORDER BY SalesAmount DESC) AS 'Rank',

SalesPersonID,

SalesPerson,

SalesAmount

FROM SalesData;

update SalesData

set SalesAmount = 500000

where SalesPersonID = 5