**MSSQL**

**Using Adventure work DB**

[SQL Server OFFSET FETCH: Limit The Number of Rows Returned (sqlservertutorial.net)](https://www.sqlservertutorial.net/sql-server-basics/sql-server-offset-fetch/)

## **DATA MANIPULATION**

* [SELECT](#_Select)
* [ORDER BY](#_Order By)
* [OFFSET FETCH](#_Fetch Top OffSet Fetch)
* [SELECT TOP](https://www.sqlservertutorial.net/sql-server-basics/sql-server-select-top/)
* [SELECT DISTINCT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-select-distinct/)
* [WHERE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-where/)
* [NULL](https://www.sqlservertutorial.net/sql-server-basics/sql-server-null/)
* [AND](https://www.sqlservertutorial.net/sql-server-basics/sql-server-and/)
* [OR](https://www.sqlservertutorial.net/sql-server-basics/sql-server-or/)
* [IN](https://www.sqlservertutorial.net/sql-server-basics/sql-server-in/)
* [BETWEEN](https://www.sqlservertutorial.net/sql-server-basics/sql-server-between/)
* [LIKE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-like/)
* [Column & Table Aliases](https://www.sqlservertutorial.net/sql-server-basics/sql-server-alias/)
* [Joins](https://www.sqlservertutorial.net/sql-server-basics/sql-server-joins/)
* [INNER JOIN](https://www.sqlservertutorial.net/sql-server-basics/sql-server-inner-join/)
* [LEFT JOIN](https://www.sqlservertutorial.net/sql-server-basics/sql-server-left-join/)
* [RIGHT JOIN](https://www.sqlservertutorial.net/sql-server-basics/sql-server-right-join/)
* [FULL OUTER JOIN](https://www.sqlservertutorial.net/sql-server-basics/sql-server-full-outer-join/)
* [Self Join](https://www.sqlservertutorial.net/sql-server-basics/sql-server-self-join/)
* [CROSS JOIN](https://www.sqlservertutorial.net/sql-server-basics/sql-server-cross-join/)
* [GROUP BY](https://www.sqlservertutorial.net/sql-server-basics/sql-server-group-by/)
* [HAVING](https://www.sqlservertutorial.net/sql-server-basics/sql-server-having/)
* [GROUPING SETS](https://www.sqlservertutorial.net/sql-server-basics/sql-server-grouping-sets/)
* [CUBE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-cube/)
* [ROLLUP](https://www.sqlservertutorial.net/sql-server-basics/sql-server-rollup/)
* [Subquery](https://www.sqlservertutorial.net/sql-server-basics/sql-server-subquery/)
* [Correlated Subquery](https://www.sqlservertutorial.net/sql-server-basics/sql-server-correlated-subquery/)
* [EXISTS](https://www.sqlservertutorial.net/sql-server-basics/sql-server-exists/)
* [ANY](https://www.sqlservertutorial.net/sql-server-basics/sql-server-any/)
* [ALL](https://www.sqlservertutorial.net/sql-server-basics/sql-server-all/)
* [UNION](https://www.sqlservertutorial.net/sql-server-basics/sql-server-union/)
* [INTERSECT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-intersect/)
* [EXCEPT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-except/)
* [Common Table Expression (CTE)](https://www.sqlservertutorial.net/sql-server-basics/sql-server-cte/)
* [Recursive CTE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-recursive-cte/)
* [INSERT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert/)
* [INSERT Multiple Rows](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert-multiple-rows/)
* [INSERT INTO SELECT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert-into-select/)
* [UPDATE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-update/)
* [UPDATE JOIN](https://www.sqlservertutorial.net/sql-server-basics/sql-server-update-join/)
* [DELETE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-delete/)
* [MERGE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-merge/)
* [PIVOT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-pivot/)
* [Transaction](https://www.sqlservertutorial.net/sql-server-basics/sql-server-transaction/)

## **Select**

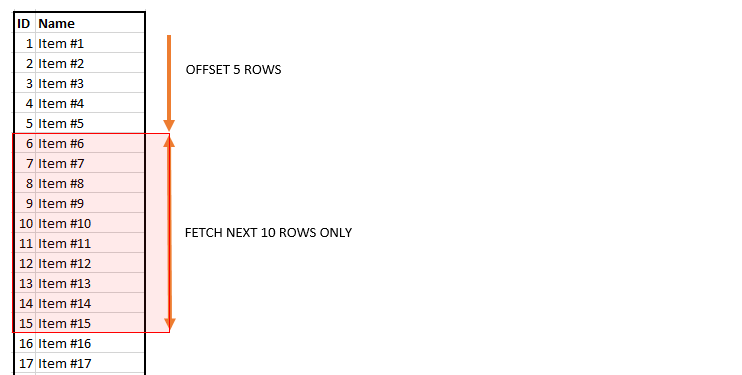
## **Order By**

select \* from Production.Product

order by ProductID desc

## **Fetch Top OffSet Fetch**

The OFFSET and FETCH clauses are the options of the [ORDER BY](https://www.sqlservertutorial.net/sql-server-basics/sql-server-order-by/) clause



select \* from Production.Product

order by ProductID asc

offset 2 rows

select \* from Production.Product

order by ProductID asc

offset 2 rows

fetch next 10 rows only

**Select Top**

select

top 2

\*

from Production.Product

select

top 2

\*

from Production.ProductListPriceHistory ph

inner join Production.Product p on p.ProductID = ph.ProductID

**Select Distinct**

select

**distinct**

top 2

ph.ListPrice,

p.Name

from Production.ProductListPriceHistory ph

inner join Production.Product p on p.ProductID = ph.ProductID

**Where**

select

top 2

ph.ListPrice,

p.Name

from Production.ProductListPriceHistory ph

inner join Production.Product p on p.ProductID = ph.ProductID

**where ph.ListPrice > 45.00 and ph.ListPrice < 50.00**

order by p.ProductID desc

**Is Null , Is Not Null**

select

top 4

ph.ListPrice,

p.Name,

ph.EndDate

from Production.ProductListPriceHistory ph

inner join Production.Product p on p.ProductID = ph.ProductID

**where ph.EndDate is Null**

select

top 4

ph.ListPrice,

p.Name,

ph.EndDate

from Production.ProductListPriceHistory ph

inner join Production.Product p on p.ProductID = ph.ProductID

**where ph.EndDate is not Null**

**And , Or**

**In**

select

top 4

ph.ListPrice,

p.Name,

ph.EndDate

from Production.ProductListPriceHistory ph

inner join Production.Product p on p.ProductID = ph.ProductID

WHERE

**ph.ListPrice IN (89.99, 109.99, 159.99)**

**Between**

select

top 4

ph.ListPrice,

p.Name,

ph.EndDate

from Production.ProductListPriceHistory ph

inner join Production.Product p on p.ProductID = ph.ProductID

**where ph.ListPrice between 89.99 and 109.99**

**Like**

**Work On String**

select

top 4

ph.ListPrice,

p.Name,

ph.EndDate

from Production.ProductListPriceHistory ph

inner join Production.Product p on p.ProductID = ph.ProductID

WHERE

**p.Name like '%p%'**

**Column and Table Aliases**

select

pPH.ListPrice,

p.Name,

pPH.EndDate

from Production.ProductListPriceHistory **pPH**

inner join Production.Product **p** on p.ProductID = pPH.ProductID

select

pPH.ListPrice as **ProductPrice**,

p.Name + '-'+p.ProductNumber as **ProductName**

from Production.ProductListPriceHistory pPH

inner join Production.Product p on p.ProductID = pPH.ProductID

**Joins**

select

pPH.ListPrice as ProductPrice,

p.Name + '-'+p.ProductNumber as ProductName

from Production.ProductListPriceHistory pPH

join Production.Product p on p.ProductID = pPH.ProductID

**Inner Join**

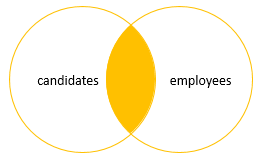
select

pPH.ListPrice as ProductPrice,

p.Name + '-'+p.ProductNumber as ProductName

from Production.ProductListPriceHistory pPH

inner join Production.Product p on p.ProductID = pPH.ProductID



**Left Join**

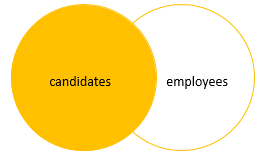
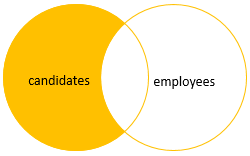
select

pPH.ListPrice as ProductPrice,

p.Name + '-'+p.ProductNumber as ProductName

from Production.ProductListPriceHistory pPH

left join Production.Product p on p.ProductID = pPH.ProductID



**Right Join**

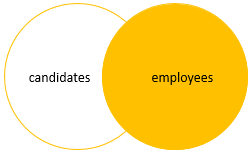
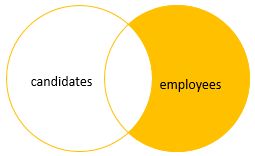
select

pPH.ListPrice as ProductPrice,

p.Name + '-'+p.ProductNumber as ProductName

from Production.ProductListPriceHistory pPH

right join Production.Product p on p.ProductID = pPH.ProductID



**Full Join**

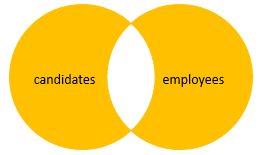
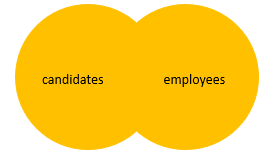
select

pPH.ListPrice as ProductPrice,

p.Name + '-'+p.ProductNumber as ProductName

from Production.ProductListPriceHistory pPH

full join Production.Product p on p.ProductID = pPH.ProductID



**Full Outer Join**

select

pPH.ListPrice as ProductPrice,

p.Name + '-'+p.ProductNumber as ProductName

from Production.ProductListPriceHistory pPH

right join Production.Product p on p.ProductID = pPH.ProductID

**Self Join (Join with Same Table)**

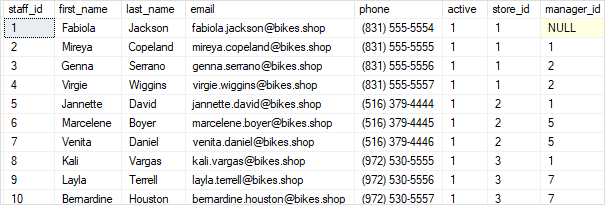
A self join allows you to join a table to itself

**--> hierarchical data**

**--> compare rows within the same table.**

A self join uses the [inner join](https://www.sqlservertutorial.net/sql-server-basics/sql-server-inner-join/) or [left join](https://www.sqlservertutorial.net/sql-server-basics/sql-server-left-join/) clause. Because the query that uses the self join references the same table,

**Exemple:**



The  staffs table stores the staff information such as id, first name, last name, and email. It also has a column named manager\_id that specifies the direct manager. For example, Mireya reports to Fabiola because the value in the manager\_id of  Mireya is Fabiola.

Tables

--[SQL Server Sample Database (sqlservertutorial.net)](https://www.sqlservertutorial.net/sql-server-sample-database/)

CREATE TABLE sales.stores (

store\_id INT IDENTITY (1, 1) PRIMARY KEY,

store\_name VARCHAR (255) NOT NULL,

phone VARCHAR (25),

email VARCHAR (255),

street VARCHAR (255),

city VARCHAR (255),

state VARCHAR (10),

zip\_code VARCHAR (5)

);

INSERT INTO sales.stores (store\_name, phone, email, street, city, state, zip\_code)

VALUES

('Store 1', '123-456-7890', 'store1@email.com', '123 Main St', 'City1', 'ST', '12345'),

('Store 2', '987-654-3210', 'store2@email.com', '456 Oak St', 'City2', 'ST', '67890');

CREATE TABLE sales.staffs (

staff\_id INT IDENTITY (1, 1) PRIMARY KEY,

first\_name VARCHAR (50) NOT NULL,

last\_name VARCHAR (50) NOT NULL,

email VARCHAR (255) NOT NULL UNIQUE,

phone VARCHAR (25),

active tinyint NOT NULL,

store\_id INT NOT NULL,

manager\_id INT,

FOREIGN KEY (store\_id)

REFERENCES sales.stores (store\_id)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (manager\_id)

REFERENCES sales.staffs (staff\_id)

ON DELETE NO ACTION ON UPDATE NO ACTION

);

SET IDENTITY\_INSERT sales.staffs ON;

INSERT INTO sales.staffs (staff\_id, first\_name, last\_name, email, phone, active, store\_id, manager\_id)

VALUES

(1, 'John', 'Doe', 'john.doe@email.com', '123-456-7890', 1, 1, NULL),

(2, 'Jane', 'Smith', 'jane.smith@email.com', '987-654-3210', 1, 2, 1),

(3, 'Bob', 'Johnson', 'bob.johnson@email.com', '555-123-4567', 1, 1, NULL),

(4, 'Alice', 'Williams', 'alice.williams@email.com', '111-222-3333', 1, 2, 2),

(5, 'Charlie', 'Brown', 'charlie.brown@email.com', '999-888-7777', 1, 1, NULL),

(6, 'Eva', 'Davis', 'eva.davis@email.com', '444-555-6666', 1, 2, 1),

(7, 'Michael', 'Taylor', 'michael.taylor@email.com', '777-666-5555', 1, 1, NULL),

(8, 'Olivia', 'Jones', 'olivia.jones@email.com', '222-333-4444', 1, 2, 2),

(9, 'Samuel', 'White', 'samuel.white@email.com', '888-999-0000', 1, 1, NULL),

(10, 'Sophia', 'Miller', 'sophia.miller@email.com', '333-444-5555', 1, 2, 1);

SET IDENTITY\_INSERT sales.staffs OFF;

select

s.staff\_id,

s.first\_name + s.last\_name as fullName,

coalesce(s\_st.first\_name,'---')

from sales.staffs s

left join sales.staffs s\_st on s\_st.staff\_id = s.manager\_id

with cte as (

select

s.staff\_id,

s.first\_name + s.last\_name as managerName,

s.manager\_id

from sales.staffs s

)

select

s.staff\_id,

s.email,

s.first\_name + s.last\_name as fullName,

coalesce(c.managerName,'-') as ManagerName

from sales.staffs s

left join cte c on c.staff\_id = s.manager\_id

**Cross Join**

**Group By**

select

ph.ListPrice,

p.Name,

ph.EndDate

from Production.ProductListPriceHistory ph

inner join Production.Product p on p.ProductID = ph.ProductID

where ph.EndDate is not Null

**group by ph.ListPrice , p.Name ,ph.EndDate**

order by p.Name desc

select

ph.ListPrice,

p.Name,

Sum(ph.ListPrice) as totalPrice

from Production.ProductListPriceHistory ph

inner join Production.Product p on p.ProductID = ph.ProductID

where ph.EndDate is not Null

group by ph.ListPrice , p.Name

**Having**

select

ph.ListPrice,

p.Name,

Sum(ph.ListPrice) as totalPrice

from Production.ProductListPriceHistory ph

inner join Production.Product p on p.ProductID = ph.ProductID

where ph.EndDate is not Null

group by ph.ListPrice , p.Name

having Name like '%C%'

**Grouping Set**

**CUBE**

**ANY**

**All**

**Union**

**Inter Section**

**Except**

**CTE**

**Insert**

**Insert Multiple Rows**

**Insert Into Select**

**Delete**

**Merge**

**PIVOT**

**Transaction**