Filtering & Joins in SQL Server

Filtering With DISTINCT

```
-- 4.3 Using SELECT DISTINCT with column having NULL
SELECT DISTINCT phone
FROM sales.customers
ORDER BY phone;
```

DISTINCT always takes first value of duplicates records
DISTINCTalways take NULL values in column as one entity.

NOTE: Both DISTINCT and GROUP BY clause reduces the number of returned rows in the result set by removing the duplicates.

---

1. OPERATOR LIKE <,>, <>/!=, IN, LIKE, BETWEEN
2. LOGICAL OPERATORS AND, OR, NOT

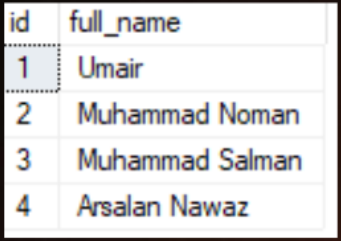
```
SELECT  product_id,  product_name, category_id, model_year, list_price
FROM production.products
WHERE list_price > 279.99 AND model_year = 2018
ORDER BY list_price ;
```

LIKE

```
-- Finding rows whose values contain a string
SELECT product_id, product_name, model_year, list_price, category_id
FROM production.products
WHERE product_name LIKE '%Townie%'
ORDER BY list_price DESC;
```

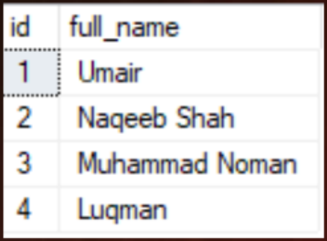
JOINS IN SQL SERVER

hr.candidates



id	full_name
1	Umair
2	Muhammad Noman
3	Muhammad Salman
4	Arsalan Nawaz

hr.emplyees



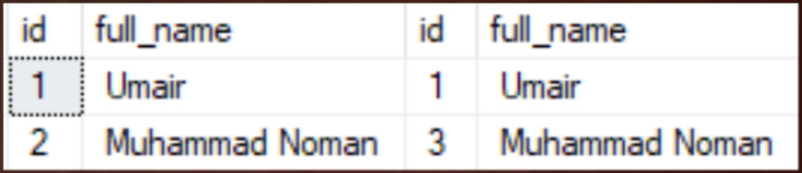
id	full_name
1	Umair
2	Naqeeb Shah
3	Muhammad Noman
4	Luqman

lets apply all joins on the two table

INNER join: It will bring the matching rows from both tables

```
SELECT  c.id, c.full_name, e.id, e.full_name
FROM hr.candidates c
INNER JOIN hr.employees e
ON c.full_name = e.full_name;
```

Result



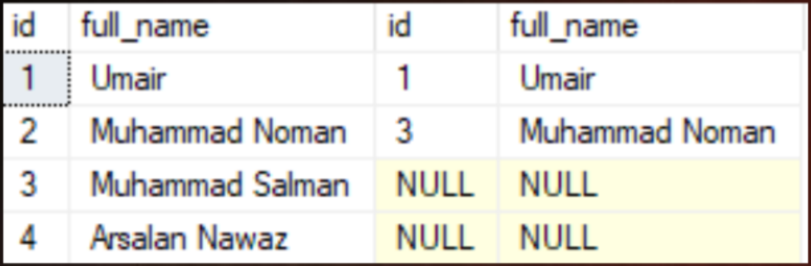
id	full_name	id	full_name
1	Umair	1	Umair
2	Muhammad Noman	3	Muhammad Noman

Returned all matchin rows in both table

JOINS IN SQL SERVER

LEFT join: It will bring the all rows from left table and matching values from right table

```
SELECT c.id, c.full_name, e.id, e.full_name
FROM hr.candidates c
LEFT JOIN hr.employees e
ON c.full_name = e.full_name;
```

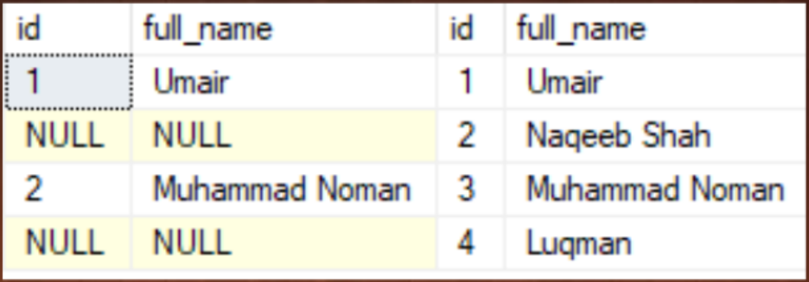


id	full_name	id	full_name
1	Umair	1	Umair
2	Muhammad Noman	3	Muhammad Noman
3	Muhammad Salman	NULL	NULL
4	Arsalan Nawaz	NULL	NULL

All Rows from left and matching from right

RIGHT join: It will bring the all rows from right table and matching values from left table

```
SELECT c.id, c.full_name, e.id, e.full_name
FROM hr.candidates c
RIGHT JOIN hr.employees e
ON c.full_name = e.full_name;
```



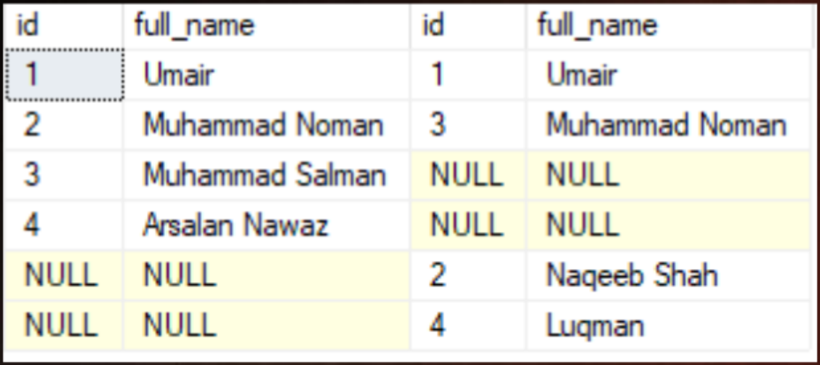
id	full_name	id	full_name
1	Umair	1	Umair
NULL	NULL	2	Naqeeb Shah
2	Muhammad Noman	3	Muhammad Noman
NULL	NULL	4	Luqman

All Rows from right and matching from left

JOINS IN SQL SERVER

OUTER/FULL join: It will bring all rows from both table.

```sql
SELECT c.id, c.full_name, e.id, e.full_name
FROM hr.candidates c
 FULL JOIN hr.employees e
 ON c.full_name = e.full_name;
```

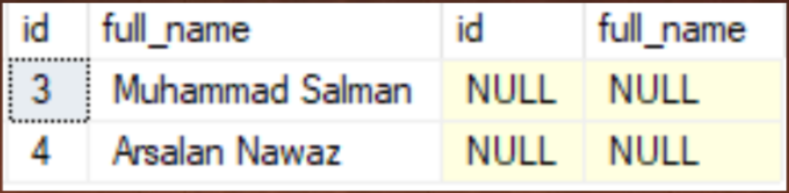


id	full_name	id	full_name
1	Umair	1	Umair
2	Muhammad Noman	3	Muhammad Noman
3	Muhammad Salman	NULL	NULL
4	Arsalan Nawaz	NULL	NULL
NULL	NULL	2	Naqeeb Shah
NULL	NULL	4	Luqman

All Rows From Left
and
All Rows From Right
But keep the
matchin rows on top

LEFT ANTI join: It will bring the all rows from left table that are not matching with right table

```sql
SELECT c.id, c.full_name, e.id, e.full_name
FROM hr.candidates c
LEFT JOIN hr.employees e
ON c.full_name = e.full_name
WHERE e.id IS NULL;
```



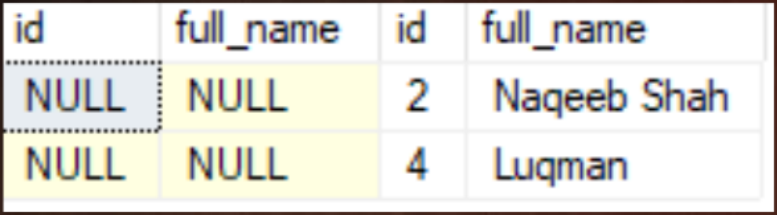
id	full_name	id	full_name
3	Muhammad Salman	NULL	NULL
4	Arsalan Nawaz	NULL	NULL

All rows from left
table that are not
matching in right
table

JOINS IN SQL SERVER

RIGHT ANTI join: It will bring the all values from right table that are not matching with left table

```
SELECT c.id, c.full_name, e.id, e.full_name
FROM hr.candidates c
RIGHT JOIN hr.employees e
ON c.full_name = e.full_name
WHERE c.id IS NULL;
```

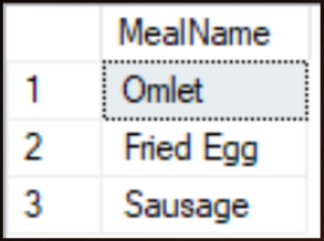


id	full_name	id	full_name
NULL	NULL	2	Naqeeb Shah
NULL	NULL	4	Luqman

All rows from right table that are not matching in left table

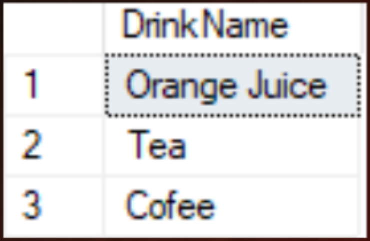
JOINS IN SQL SERVER

cross_join.Meals



	MealName
1	Omlet
2	Fried Egg
3	Sausage

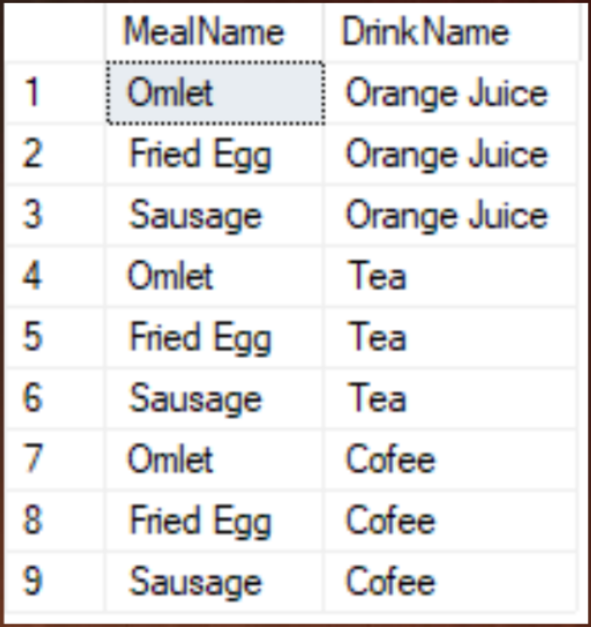
cross_join.Drinks



	DrinkName
1	Orange Juice
2	Tea
3	Cofee

Cross Join in SQL produces a result set that contains the cartesian product of two or more tables. Cross join is also called a Cartesian Join.

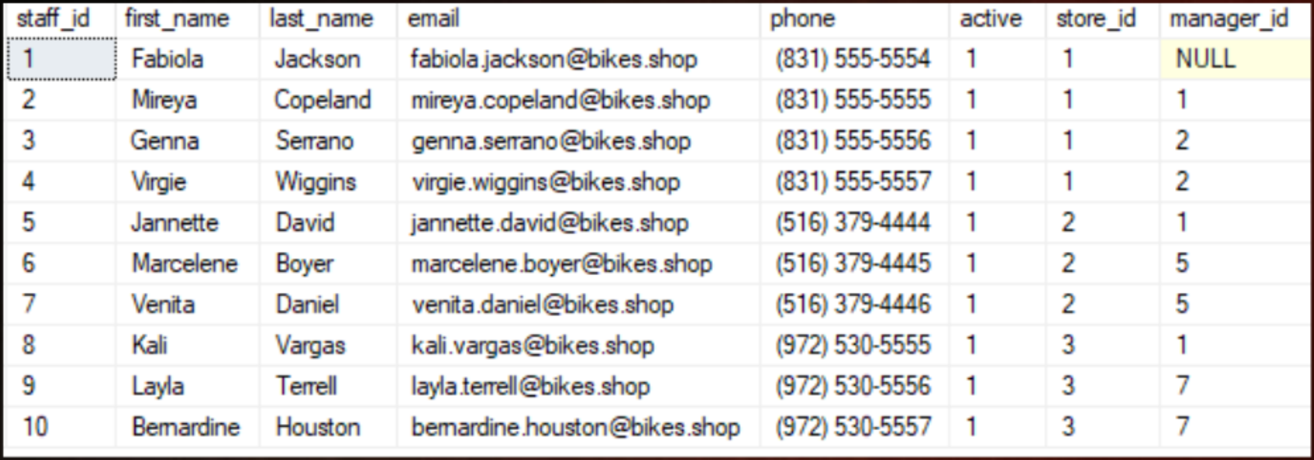
```
SELECT *
FROM cross_join.Meals
CROSS JOIN cross_join.Drinks;
```



	MealName	DrinkName
1	Omlet	Orange Juice
2	Fried Egg	Orange Juice
3	Sausage	Orange Juice
4	Omlet	Tea
5	Fried Egg	Tea
6	Sausage	Tea
7	Omlet	Cofee
8	Fried Egg	Cofee
9	Sausage	Cofee

JOINS IN SQL SERVER

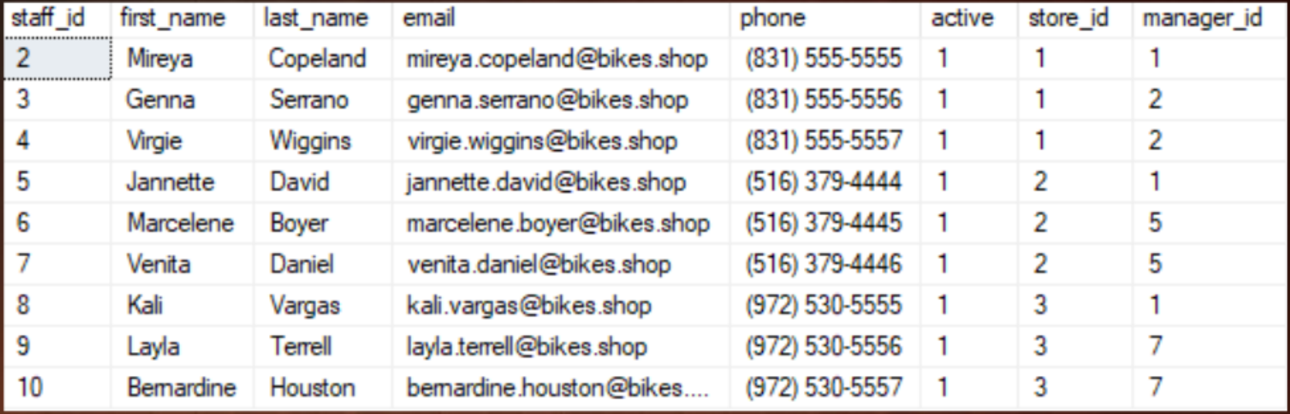
SELF join: A self join is regular join in which a table is joined to itself.



staff_id	first_name	last_name	email	phone	active	store_id	manager_id
1	Fabiola	Jackson	fabiola.jackson@bikes.shop	(831) 555-5554	1	1	NULL
2	Mireya	Copeland	mireya.copeland@bikes.shop	(831) 555-5555	1	1	1
3	Genna	Serrano	genna.serrano@bikes.shop	(831) 555-5556	1	1	2
4	Virgie	Wiggins	virgie.wiggins@bikes.shop	(831) 555-5557	1	1	2
5	Jannette	David	jannette.david@bikes.shop	(516) 379-4444	1	2	1
6	Marcelene	Boyer	marcelene.boyer@bikes.shop	(516) 379-4445	1	2	5
7	Venita	Daniel	venita.daniel@bikes.shop	(516) 379-4446	1	2	5
8	Kali	Vargas	kali.vargas@bikes.shop	(972) 530-5555	1	3	1
9	Layla	Terrell	layla.terrell@bikes.shop	(972) 530-5556	1	3	7
10	Bernardine	Houston	bernardine.houston@bikes.shop	(972) 530-5557	1	3	7

```
SELECT *
FROM sales.staffs AS T1
INNER JOIN sales.staffs AS T2
ON T2.staff_id = T1.manager_id;
```

**All Rows from left
and
matching from right**



staff_id	first_name	last_name	email	phone	active	store_id	manager_id
2	Mireya	Copeland	mireya.copeland@bikes.shop	(831) 555-5555	1	1	1
3	Genna	Serrano	genna.serrano@bikes.shop	(831) 555-5556	1	1	2
4	Virgie	Wiggins	virgie.wiggins@bikes.shop	(831) 555-5557	1	1	2
5	Jannette	David	jannette.david@bikes.shop	(516) 379-4444	1	2	1
6	Marcelene	Boyer	marcelene.boyer@bikes.shop	(516) 379-4445	1	2	5
7	Venita	Daniel	venita.daniel@bikes.shop	(516) 379-4446	1	2	5
8	Kali	Vargas	kali.vargas@bikes.shop	(972) 530-5555	1	3	1
9	Layla	Terrell	layla.terrell@bikes.shop	(972) 530-5556	1	3	7
10	Bernardine	Houston	bernardine.houston@bikes....	(972) 530-5557	1	3	7