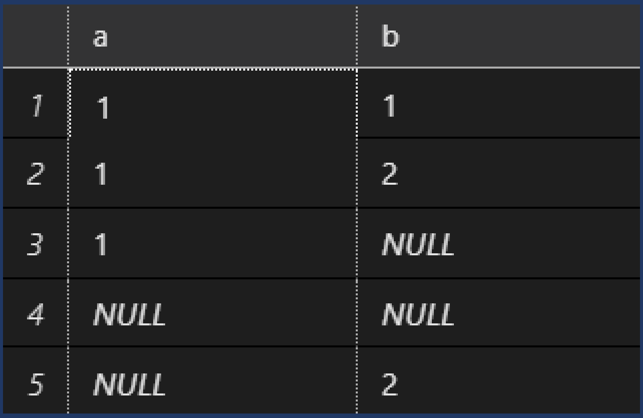
bite-sized.sql

SQL: NULL-safe comparisons

bite-sized.sql

Setup

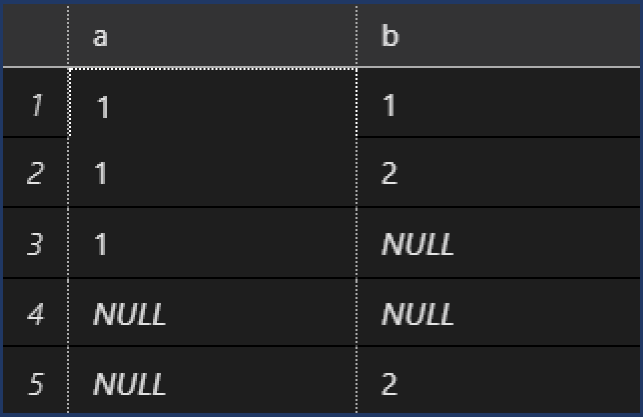
```
DROP TABLE IF EXISTS null_comparisons;

CREATE TABLE null_comparisons
(
    a int,
    b int
);

INSERT INTO null_comparisons
    (a,b)
VALUES
    (1, 1),
    (1, 2),
    (1, null),
    (null, null),
    (null, 2);
```



	a	b
1	1	1
2	1	2
3	1	NULL
4	NULL	NULL
5	NULL	2

bite-sized.sql

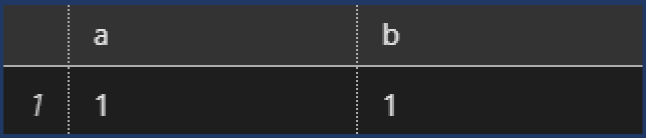
Select rows where a = b



	a	b
1	1	1
2	1	2
3	1	*NULL*
4	*NULL*	*NULL*
5	*NULL*	2

This row is not returned

```
SELECT *
FROM null_comparisons
WHERE a = b;
```

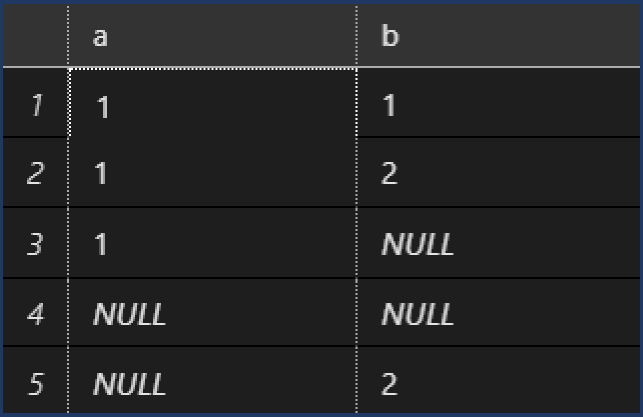


	a	b
1	1	1

Let’s find where a is not equal to b

bite-sized.sql

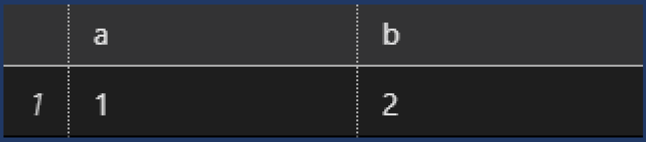
Select rows where a <> b



	a	b
1	1	1
2	1	2
3	1	NULL
4	NULL	NULL
5	NULL	2

Both of these rows
are not returned

```sql
SELECT *
FROM null_comparisons
WHERE a <> b;
```

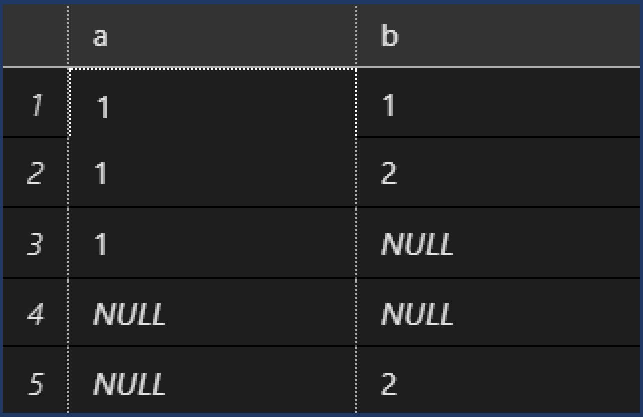


	a	b
1	1	2

flexyourdata.com | youtube.com/@flexyourdata | linkedin.com/in/owenhprice

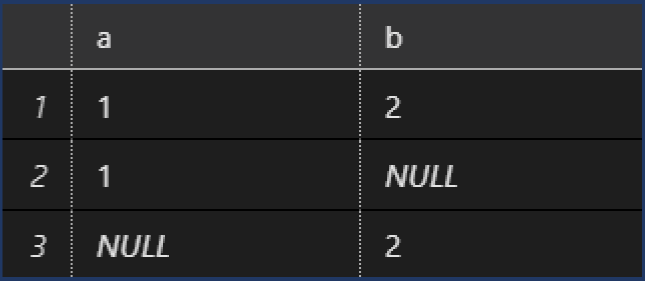
bite-sized.sql

The vendor-independent long-winded way



	a	b
1	1	1
2	1	2
3	1	NULL
4	NULL	NULL
5	NULL	2

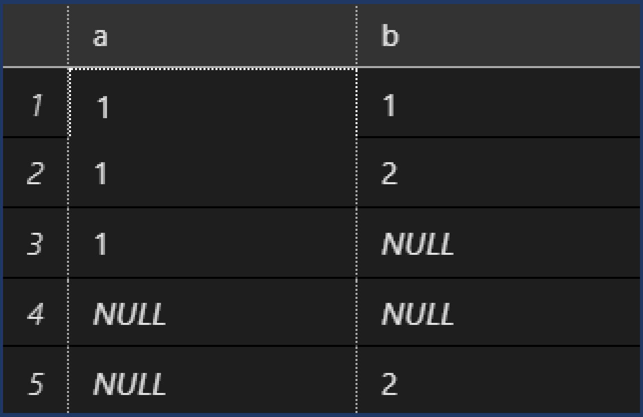
```
SELECT *
FROM null_comparisons
WHERE (a <> b)
OR (a IS NULL AND b IS NOT NULL)
OR (a IS NOT NULL AND b IS NULL);
```



	a	b
1	1	2
2	1	NULL
3	NULL	2

bite-sized.sql

SQL Server versions < 2022

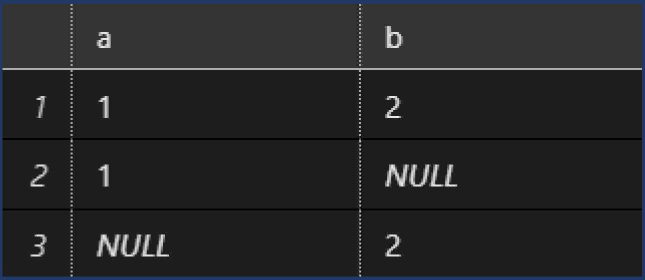


	a	b
1	1	1
2	1	2
3	1	NULL
4	NULL	NULL
5	NULL	2

The value you select for the second parameter of IFNULL or COALESCE must not be a legitimate value and it must be type-compatible with the other side of the comparison

```sql
SELECT *
FROM null_comparisons
WHERE ISNULL(a,0) <> ISNULL(b,0);
```

ISNULL returns the first argument if it's not NULL and the second argument if the first argument is NULL



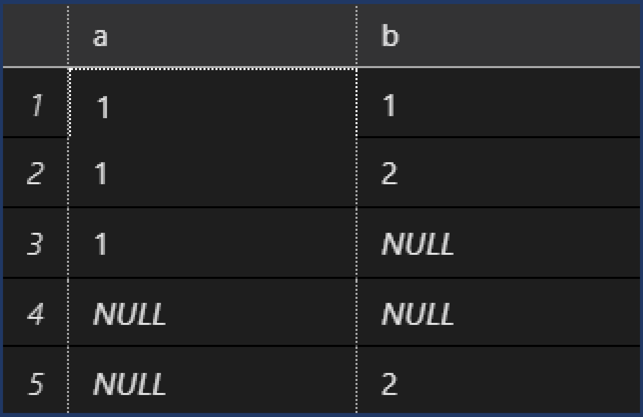
	a	b
1	1	2
2	1	NULL
3	NULL	2

We can also use COALESCE in the same way

e.g. COALESCE(a,0)

bite-sized.sql

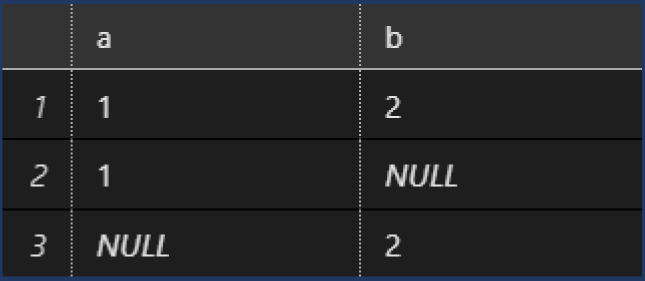
SQL Server 2022+ and PostgreSQL



	a	b
1	1	1
2	1	2
3	1	NULL
4	NULL	NULL
5	NULL	2

```sql
SELECT *
FROM null_comparisons
WHERE a IS DISTINCT FROM b;
```

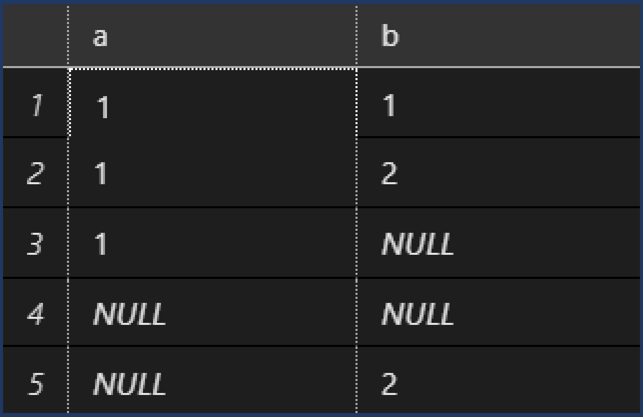
IS DISTINCT FROM is a NULL-safe inequality predicate. We can also use IS NOT DISTINCT FROM for a NULL-safe equality predicate.



	a	b
1	1	2
2	1	NULL
3	NULL	2

bite-sized.sql

MySQL

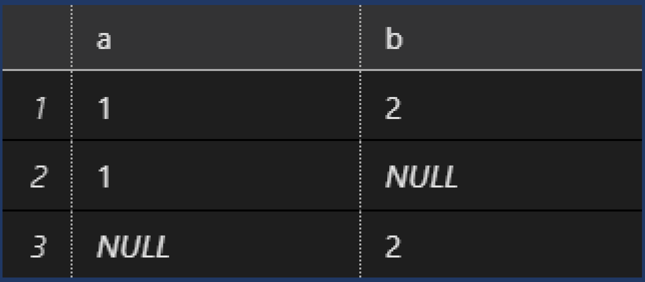


	a	b
1	1	1
2	1	2
3	1	NULL
4	NULL	NULL
5	NULL	2

! In MySQL, we can also use IFNULL(a,0) <> IFNULL(b,0) in a similar way to SQL Server, but NOT with <=> is more safe as we don't need to specify a default value for NULLs

```
SELECT *
FROM null_comparisons
WHERE NOT a <=> b;
```

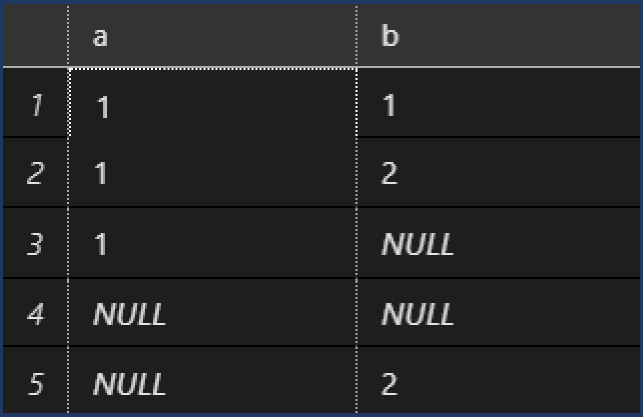
While the <=> operator looks like a null-safe "not equals", it is actually a null-safe "equals", so we precede it with NOT here to get "not equals".



	a	b
1	1	2
2	1	NULL
3	NULL	2

bite-sized.sql

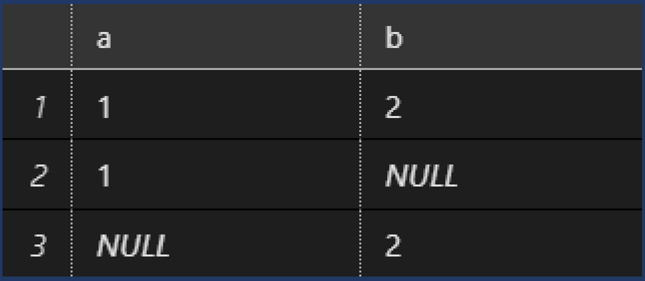
Oracle - 1



	a	b
1	1	1
2	1	2
3	1	NULL
4	NULL	NULL
5	NULL	2

```
SELECT *
FROM null_comparisons
WHERE NVL(a,0) <> NVL(b,0);
```

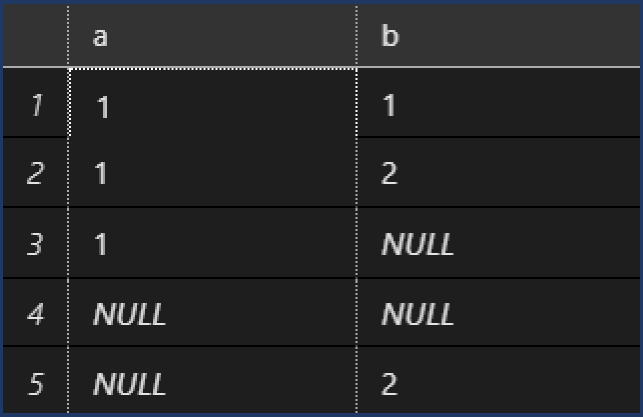
Similarly to T-SQL's ISNULL and MySQL's IFNULL, we must be careful when choosing a default value.



	a	b
1	1	2
2	1	NULL
3	NULL	2

bite-sized.sql

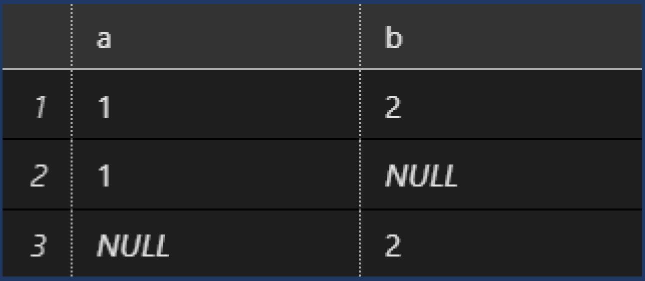
Oracle - 2



	a	b
1	1	1
2	1	2
3	1	NULL
4	NULL	NULL
5	NULL	2

```
SELECT *
FROM null_comparisons
WHERE DECODE(a,b,'equal','unequal') = 'unequal';
```

DECODE compares the first and second arguments. If they're equal, it returns the third argument, if they're unequal, it returns the fourth argument if supplied, or NULL if there's no fourth argument. The third and fourth arguments can be any value.



	a	b
1	1	2
2	1	NULL
3	NULL	2

To the guy who invented NULL
THANKS FOR NOTHING!
imgflip.com