Union, Minus, and Intersect

```
create table my_brick_collection ( colour varchar2(10), shape varchar2(10), weight integer );

create table your_brick_collection ( height integer, width integer, depth integer, colour varchar2(10), shape varchar2(10) );


insert into my_brick_collection values ( 'red', 'cube', 10 );

insert into my_brick_collection values ( 'blue', 'cuboid', 8 );

insert into my_brick_collection values ( 'green', 'pyramid', 20 );

insert into my_brick_collection values ( 'green', 'pyramid', 20 );

insert into my_brick_collection values ( null, 'cuboid', 20 );


insert into your_brick_collection values ( 2, 2, 2, 'red', 'cube' );

insert into your_brick_collection values ( 2, 2, 2, 'blue', 'cube' );

insert into your_brick_collection values ( 2, 2, 8, null, 'cuboid' );
```

The set operators union, intersect, and minus allow you to combine many tables into one.

The union operator combines two or more tables into a single result set. To use it, the tables must have the same number of columns with matching data types in each position.

The brick collection tables have different columns. So combining these with select * leads to an error : ORA-01789: query block has incorrect number of result columns

```
select * from my_brick_collection union

select * from your_brick_collection;
```

To resolve this, select the common columns. Here that's the colour and shape. So this query returns a list of all the (colour, shape) values in the tables:

```
select colour, shape from my_brick_collection
union
select colour, shape from your_brick_collection;
```

Distinct

There are two red cube rows in the tables, one in each table. But union only displays one!

This is because union applies the distinct operator to your results. This discards duplicate rows. So your results only have one row for each set of column values.

You can use distinct by placing it after select and before the column list. This squashes out duplicates. So you only get one row for each set of values in your columns.

For example, there are two green pyramid rows in my collection. If you do a distinct *, you only get one copy of this brick:

```
select distinct * from my_brick_collection;
```

You can also use distinct on a subset of a table's columns. For example, to get one row for each shape in your collection, select "distinct shape":

```
select distinct shape from your_brick_collection;
```

Union All

Often when combining tables you want to see all the rows. Including duplicates. Not the list of distinct values.

You can do this by sticking all after union:

```
select colour, shape from my_brick_collection
union all
select colour, shape from your_brick_collection;
```

A standard union is the same as the following:

```
select distinct * from (
  select colour, shape from my_brick_collection
  union all
  select colour, shape from your_brick_collection
);
```

The distinct operator adds an extra sorting step to your SQL. So in most cases you'll want to use union all. Save plain union for when you know you want to remove duplicate rows.

Try It!

Complete this query to return a list of all the colours in the two tables. Each colour must only appear once:

```
select colour from my_brick_collection

select colour from your_brick_collection
order by colour;
```

The output of this query should be:

```
COLOUR
blue
green
red
<null>
```

Complete the following query to return a list of all the shapes in both tables. There must show one row for each row in the source tables:

```
select shape from my_brick_collection

select shape from your_brick_collection
order  by shape;
```

This query should return the following rows:

```
SHAPE
cube
cube
cube
cuboid
cuboid
cuboid
pyramid
pyramid
```

Minus

Oracle Database includes an operator that implements set difference: minus

All you need to do is select the relevant columns from each table with minus between them.

```
select colour, shape from my_brick_collection
minus
```

```
select colour, shape from your_brick_collection;
```

COLOUR	**SHAPE**
blue	cuboid
green	pyramid

Like union, minus applies a distinct to the output. There are two green pyramids in my collection not in yours. But minus only returns one of these:

```
select colour, shape from my_brick_collection
minus
select colour, shape from your_brick_collection
```

Intersect

There is also an operator to find the common values: intersect

You use this in the same way as union and minus: place it between a select from each table:

```
select colour, shape from your_brick_collection
intersect
select colour, shape from my_brick_collection;
```

As with minus, the database considers null values to be the same and applies a distinct operator to the results.

Try It!

Complete the following query to return a list of all the shapes in my collection not in yours:

```
select shape from my_brick_collection

select shape from your_brick_collection;
```

This should return the following row:

```
SHAPE
pyramid
```

Complete the following query to return a list of all the colours that are in both tables:

select colour from my_brick_collection

select colour from your_brick_collection
order by colour;

This should return the following rows:

COLOUR
blue
red
<null>

→
Extragerea datelor folosind Subquery, subinterogari

--top n queries/primele n inregistrari

```
Select last_name, salary
from (select last_name, salary from employees order by salary)
where rownum<6;

select * from employees
FETCH FIRST 10 ROWS ONLY;
```

-- multiple row query
```
Select last_name, salary from employees where salary > (select avg(salary) from employees);
```

Exercitii:

Utilizand tabelele emp si dept sa se obtina urmatoarele informatii :

- ➔ 1. Afisare job, nume, nr. departament si salariu, pt. angajatii cu salariul minim pe job

- ➔ 2. afisare numele angajatului care are a treia litera a numelui 'A'

→3. afisare nume, data angajare si data revizuire salariu(dupa 6 luni de la angajare)

- ➔ 4. Utilizati sintaxa case pentru a afisa un mesaj pentru fiecare luna din an sau sa afisati luna in lb. romana. Sa se afiseze numele angajatului, data angajarii si mesajul aferent lunii.

→5. Sa se afiseze data de angajare cea mai veche si cea mai noua
→6.

Utilizand diversele variante de join-uri, prezentate si in link-ul urmatorul, faceti diverse legaturi intre tabele employees si departments si observati rezultatele obtinute.

https://www.techonthenet.com/oracle/joins.php