* Nested subquery -
SQL provides a mechanism for nesting subque
A subquery is a select ___ from ___ where
expression i.e. nested within other query.
A common use of subquery is to perform a
test for ~~set~~ membership, make set compari-
sons and determine its set cardinality.

• Set membership -
The 'in' connective test for set membershi
where the set is the collection of values
produced by a select clause. The 'not in'
connective tests for an absence of set mem-
bership.

e.g. Find salary of employee from employee rela-
tion whose name is Bhushan or Ahire or
Bhadane.

```
SQL> select salary.
     from employee
     where name in('Bhushan', 'Ahire', 'Bhadane
```

ex. Find all customers who have both loan and
account.
We begin finding all customers ~~cust~~ name
which have accounts and we write the
subquery as

```
SQL> select cname
     from depositor;
```

But we need to find those customers
who are borrowers that means they have
a loan and they also appear in the list

of depositor having accounts obtain in the subquery. The resulting complete query will be

SQL>
```
select cname from borrower
where cname in (
                select cname
                from depositor
                );
```

The above ex. shown can be written in several way in SQL.

We can use not in construct in similar ways

e.g. find all customers who do not have a loan at Bank But have an account at the Bank.

SQL>
```
select cname
from ~~borrower~~ depositor
where cname not in (
                select cname
                from ~~depositor~~ borrower
                );
```

ex. Find all customers who have ~~account~~ loan at bank whose names are neither smith nor John.

SQL>
```
select cname
from ~~depositor~~ borrower
where cname not in ('Smith', 'John'); ↵
```

• Set comparison –

Nested subqueries are used to compare sets. SQL uses various comparison operators such as ~~less th~~ <, <=, >, >=, =, <>, any, 86)

all and some to compare sets. SQL allows following set comparison operators

| | |
|---|---|
| < some | - less than atleast one. |
| <= some | - less than or equal to atleast o |
| > some | - greater than atleast one |
| >= some | - greater than or equal to atleast one. |
| = some | - equal to atleast One. |
| <> some | - not equal to atleast one. |

consider the following relation.
Branch ( Bname, Bcity, Assets)
ex - Find Branch name that have assets greater than those of atleast one branc located in Dhule city.

```
SQL> select Bname
from Branch
where Assets > some (select Assets from
                     Branch where
                     Bcity = 'Dhule'); ↵
```

The subquery select Assets from Branch where Bcity = 'Dhule' generates the set of all Assets value for all branches located in Dhule city. It is verified that '= some' is identical to 'in' but ~~not~~ ~~a~~ '<> some' is not identical to 'not in'.
SQL also allows set comparison operation using all:

| | |
|---|---|
| < all | - less than all |
| <= all | - less than or equal to all |

>all - greater than all
>=all - greater than equal to all
=all - equal to all
<>all - not equal to all.

eg. Find the names of all branches that have an Asset value greater than all branches located at Dhule city.

Ans.
```
select Bname
from Branch
where Assets >all ( select Assets
                    from Branch
                    where Bcity = 'Dhule');
```