

Structured Query Language (SQL)

Other Statements

SQL

- IN construct

- (a) IN (a, b, c) returns TRUE

- (d) IN (a, b, c) returns FALSE

- Example:

- Find all Customers who have a loan and an account at France Branch

- We solved this query before using the Intersect operation.

- First we need to find out the name of Customers who have an account at the France Branch. Call this set S1

- (select cname
from Account
where lower(bname) = 'france')

SQL

- Example continued

- Second we need to find the list of all Loaners from France whose name appears in set S1

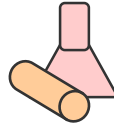
```
select cname  
from Loan  
where lower(bname) = 'france' AND  
       cname IN S1;
```

SQL

- The Complete statement:

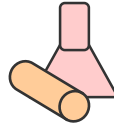
```
select cname
from Loan
where lower(bname) = 'france' AND
       cname IN
           (select cname
            from Account
            where lower(bname) = 'france');
```

SQL



- Example:
 - Find the name of all customers who have a loan and an account **in branches** in Minnetonka using the IN construct
 - **Note:** the account and the loan do not need to be in the same branch in Minnetonka

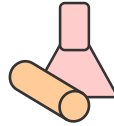
SQL



- Example:
 - Find the name of all customers who have a loan and an account **in branches** in Minnetonka using the IN construct
 - **Note:** the account and the loan do not need to be in the same branch in Minnetonka

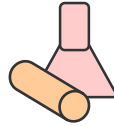
```
select cname
from Loan, Branch
where lower(bcity) = 'minnetonka' AND
loan.bname = branch.bname AND
Cname IN
    (select cname
     from Account, Branch
     where lower(bcity) = 'minnetonka' AND
     account.bname = branch.bname);
```

SQL



- Example:
 - Find the name of all customers who have a loan and an account **in the same branch** in Minnetonka using the IN construct
 - **Note:** the account and the loan must be in the same branch in Minnetonka

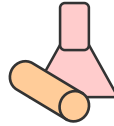
SQL



- Example:
 - Find the name of all customers who have a loan and an account **in the same branch** in Minnetonka using the IN construct
 - **Note:** the account and the loan must be in the same branch in Minnetonka

```
select cname
from Loan, Branch
where lower(bcity) = 'minnetonka' AND
loan.bname = branch.bname AND
Cname IN
    (select cname
     from Account, Branch
     where lower(bcity) = 'minnetonka' AND
      account.bname = branch.bname AND
      account.bname = loan.bname);
```

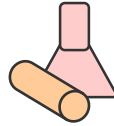

SQL



- Example:
 - We can simplify the statement since when `account.bname = loan.bname` we do not need the `lower(bcity) = 'minnetonka'`

```
select cname
from Loan, Branch
where lower(bcity) = 'minnetonka' AND
loan.bname = branch.bname AND
Cname IN
    (select cname
     from Account
     where account.bname = loan.bname);
```

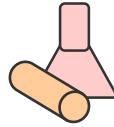
SQL



- Another approach:
 - Instead of checking the customer name only in the IN statement, we check the combination of cname, bname

```
select cname
from Loan, Branch
where lower(bcity) = 'minnetonka' AND
loan.bname = branch.bname AND
(cname, loan.bname) IN
(select cname, account.bname
from Account, Branch
where lower(bcity) = 'minnetonka' AND
account.bname = branch.bname AND
account.bname = loan.bname);
```

SQL

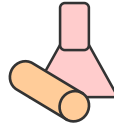


- Since we use the combination of cname and bname, we can simplify the answer:

```
select cname
from Loan, Branch
where lower(bcity) = 'minnetonka' AND
loan.bname = branch.bname AND
(cname, loan.bname) IN
    (select cname, account.bname
     from Account, Branch
     where lower(bcity) = 'minnetonka' AND
     account.bname = branch.bname) AND
     account.bname = loan.bname);
```

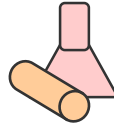
```
select cname
from Loan, Branch
where lower(bcity) = 'minnetonka' AND
loan.bname = branch.bname AND
(cname, loan.bname) IN
    (select cname, account.bname
     from Account);
```

SQL



- Example:
 - Find the name of all customers who have a loan and an account **in branches** in Minnetonka using the **EXISTS** construct
 - **Note:** the account and the loan do not need to be in the same branch in Minnetonka

SQL



- Example:

- Find the name of all customers who have a loan and an account **in branches** in Minnetonka using the **EXISTS** construct
 - **Note:** the account and the loan do not need to be in the same branch in Minnetonka

```
select account.cname
from account, branch
where    lower(branch.bcity) = 'minnetonka' AND
        branch.bname = account.bname AND
EXISTS
        (select loan.cname
         from loan , branch
         where    lower(branch.bcity) = 'minnetonka' AND
                 branch.bname = loan.bname AND
                 loan.cname = account.cname
        );
```

SQL

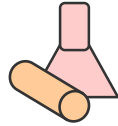
- Find all Customers who have a loan at France Branch but NOT an account there.

```
select cname
from Loan
where lower(bname) = 'france' AND
       cname NOT IN
           (select cname
            from Account
            where lower(bname) = 'france'
           );
```

SQL

- SQL table aliases
- Used to represent a table
- Are defined in the 'From' clause of the select statement
- Example:
 - Find the name of all Customers who have an account at a Branch at which Jones has an account

SQL



- Using the IN statement

select cname

from Account

where bname IN

(select bname

from Account


where lower(cname) = 'jones');

SQL

- Using table aliases

```
select T.cname  
from Account S, Account T  
where lower(S.cname) = 'jones' AND  
       S.bname = T.bname AND  
       lower(T.cname) != 'jones';
```

Account Table aliased S

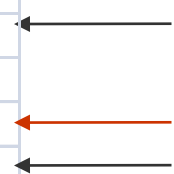


BNAME	CNAME
c1	Johnson
b1	Jones
b1	Smith
b2	Jones
b2	Rogers
b1	Cook

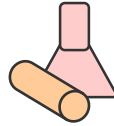
Account Table aliased T

BNAME	CNAME
c1	Johnson
b1	Jones
b1	Smith
b2	Jones
b2	Rogers
b1	Cook

Results



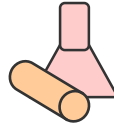
SQL



- Try it out

Print the balance and the branch name of all accounts of a customer if the customer has an account with a balance of equal to or more than 1000.

SQL



- Try it out

Print the balance and the branch name of all accounts of a customer if the customer has an account with a balance of equal to or more than 1000.

```
-- The first table (S) is to find the account with
-- bal => 1000
-- The second table (T) is to print bal and bname
--
select Distinct T.bname, T.bal
from Account S, Account T
where S.bal >= 1000 AND
      S.cname = T.cname;
```

SQL

- Use of ANY/SOME or ALL
 - $(12) > \text{ANY/SOME}(30, 10, 13)$ is TRUE
 - $(12) > \text{ALL}(30, 10, 13)$ is FALSE
 - $(35) > \text{ALL}(30, 10, 13)$ is TRUE
- Find all Branches that have assets greater than assets at some Branches in Edina

```
select bname
from Branch
where assets > ANY
(select assets
from Branch
where lower(bccity) = 'edina');
```

SQL

- Find all Branches that have assets greater than assets at all Branches in Edina

```
select bname
from Branch
where assets > all
  (select assets
   from Branch
   where lower(bccity) = 'edina');
```

SQL

- Set Containment
 - ANY/SOME or ALL, etc. allow comparison of one value against the members of a SET.
 - Sometimes we need to perform a set containment operation, something equivalent to the following that:
 - Returns true for (a, b, c) CONTAINS (a, b)
 - Returns false for (a, b, c) CONTAINS (a, d)

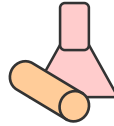
SQL

- Example - Find all Customers who have an account at ALL Branches in Edina
 - First, we have to find the list of all Branches for every Customers accounts (S1)
 - Then we have to find the name of all Branches in Edina (S2)
 - For every Customer, we have to check to see if S1 contains S2. If it does that Customer is one of the answers

SQL

- SQL DOES NOT HAVE an equivalent operator for “Contains”
- We will look at a couple of different ways of writing equivalent queries in

SQL



- Try it:
 - Step 1:
 - For every customer, find the name of all branches for this customer's accounts. Call this set S1
 - Step 2:
 - Find branches in Edina. Call this set S2
 - Step 3:
 - Check if there is a branch in S2 that is not in S1 for each customer. If that is the case, then this customer IS NOT one of the answer – throw this customer away. What is left gives you the answer.

SQL

- A list of customers and their account branches – S1

Select **a1.cname, a1.bname** from account a1

Where exists

```
( select a3.bname  
  from account a3  
  where a1.cname = a3.cname  
);
```

SQL

- List of branches in Edina – S2

```
select bname from branch  
where lower(bcity) = 'edina' ;
```

SQL

- Tie the two pieces together using the NOT EXISTS and NOT IN

```
Select distinct a1.cname from account a1
where NOT EXISTS (
    select bname from branch
    where lower(bcity) = 'edina'
    and bname NOT IN
        (select a3.bname
         from account a3
         where a1.cname = a3.cname
        )
);
```

SQL

- We can also tie the pieces together using NOT EXISTS and MINUS

```
Select distinct a1.cname from account a1
where NOT EXISTS (
    (select bname from branch
     where lower(bcity) = 'edina')
  MINUS
  (select a3.bname
   from account a3
   where a1.cname = a3.cname )
);
```

SQL

- Will the following also work? If yes, why? If not, why?

```
Select distinct a1.cname from account a1
where NOT EXISTS (
    (select a3.bname
     from account a3
     where a1.cname = a3.cname )
MINUS
(select bname from branch
 where lower(bcity) = 'edina')
);
```