Example of Beers

#	name	manf
0	Blue	Labatt's
1	Bud	Anheuser-Busch
2	Bud Light	Anheuser-Busch
3	Amnesiac	Phillips
4	Chimay Blue	Chimay
5	Blue Buck	Phillips

any operator

- x = **any** (<subquery>)
  - is a boolean condition that evaluates to true if and only if x equals at least one tuple in the subquery result
  - The operator can be a relational operator
- For instance: x >= **any** (<subquery>)
  - Is true if x is not the uniquely smallest tuple produced by the subquery
  - Implied here is that the tuple has single attribute

all operator

- x <> **all**(<subquery>)
  - is a boolean condition that evaluates to true if and only if for every tuple t in the relation, x is not equal to t
  - Put differently: x is not in the subquery result
  - As with **any**, the operator can be a relational operator
- For instance: x >= **all** (<subquery>)
  - Is true if there is no tuple larger than x in the subquery result
  - Implied here is that the tuple has single attribute

Example

- Consider the relation:
  - Sells(pub, beer, price)
- Want an answer to the following:
  - Find the beer(s) sold for the highest price

```
select beer
from   Sells
where  price >= all (select price
                     from   Sells);
```

Union; intersection; difference

- Set operations available in the RA are also available in SQL
- They are usually expressed using subqueries
- $\cup$: (<subquery1> **union** <subquery2>)
- $\cap$: (<subquery1> **intersect** <subquery2>)
- $-$: (<subquery1> **except** <subquery2>)

Example

- Consider the relations:
  - Likes(patron, beer)
  - Sells(pub, beer, price)
  - Frequents(patron, pub)
- Want an answer to the following:
  - Find the patrons and beers such that (a) the patron likes the beer and (b) the patron frequents at least one pub that sells the beer.

Example

Note that we transform a stored table into a subquery result using the parentheses!

```
(select * from Likes)
    intersect
(select patron, beer
 from   Sells, Frequents
 where  Frequents.pub = Sells.pub);
```