COMP 3311
DATABASE MANAGEMENT SYSTEMS

LECTURE 9 EXERCISES
STRUCTURED QUERY LANGUAGE (SQL)

BOOK STORE RELATIONAL SCHEMA

Book(bookId, title, subject, quantityInStock, price, *authorId*)

Author(authorId, firstName, lastName)

Customer(customerId, firstName, lastName)

BookOrder(orderId, *customerId*, orderYear)

OrderDetails(*orderId, bookId*, quantity)

Attribute names in italics are foreign key attributes.

Assumptions

- Each author has authored at least one book in the store.
- Each book has exactly one Author.
- Each order is made by exactly one customer and has one or more associated tuples in OrderDetails (e.g., one order may contain several different books).

EXERCISE 1

Find all distinct book titles of the author whose last name is Piper.

Relational Algebra

$$\pi_{\text{title}}(\sigma_{\text{lastName='Piper'}}(\text{Author JOIN Book}))$$

$$\pi_{\text{title}}((\sigma_{\text{lastName='Piper'}}\text{Author}) \text{ JOIN Book})$$

SQL

```
select distinct title
from Book natural join Author
where lastName='Piper';
```

EXERCISE 2

Find the last name and first name of all authors who wrote books in both the subjects of Art and Business.

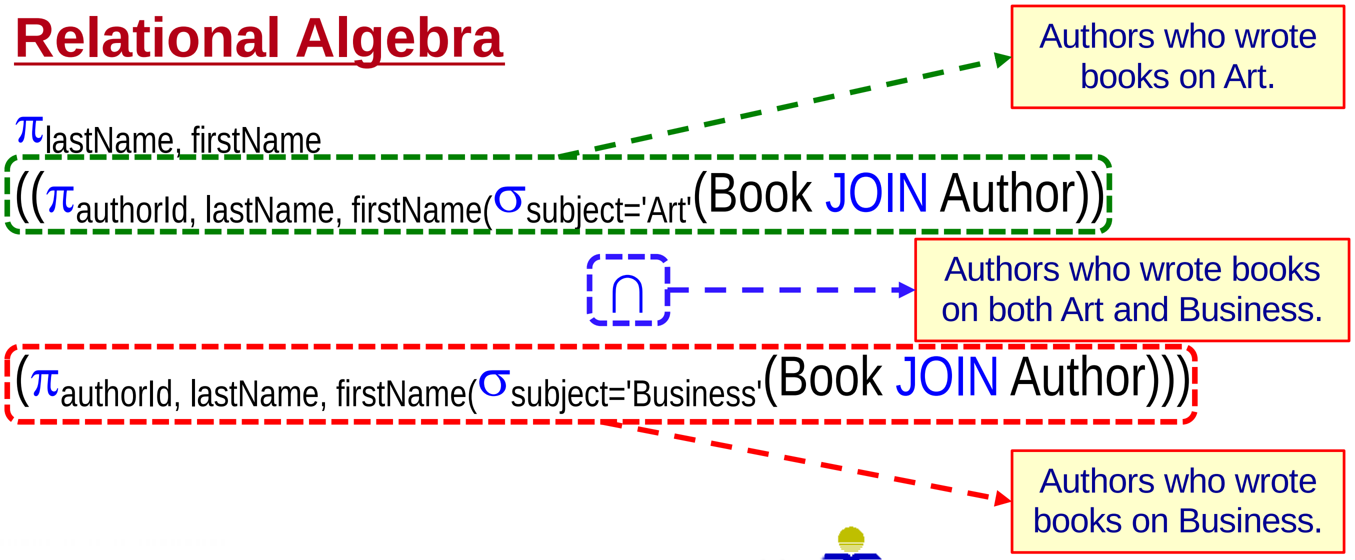
Can we say ⟹ where subject='Art' **and** subject='Business'? **No. Why?**

☞ **Selects nothing.**

Can we say ⟹ where subject='Art' **or** subject='Business'? **No. Why?**

☞ **Selects authors who wrote either Art or Business books, but not necessarily on both subjects.**

Relational Algebra



$\pi_{lastName, firstName}$

$((\pi_{authorId, lastName, firstName}(\sigma_{subject='Art'}(Book\ JOIN\ Author))$

$\cap$

$(\pi_{authorId, lastName, firstName}(\sigma_{subject='Business'}(Book\ JOIN\ Author)))$

Authors who wrote books on Art.

Authors who wrote books on both Art and Business.

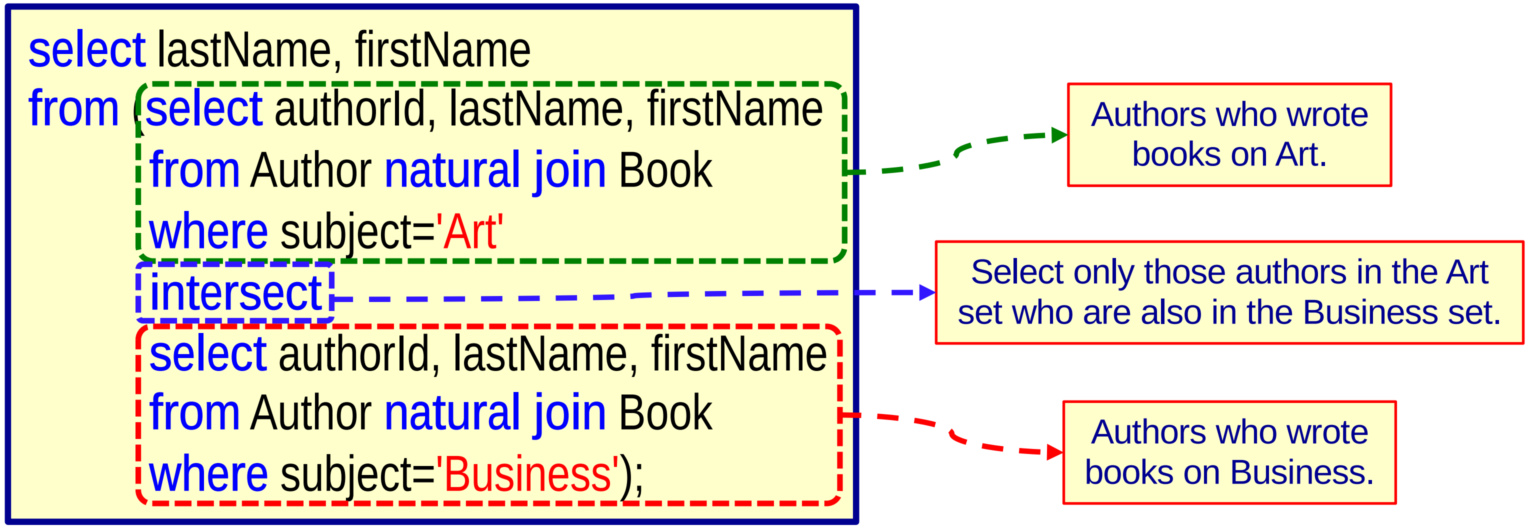
Authors who wrote books on Business.

Book(bookId, title, subject, quantityInStock, price, *authorId*) Author(authorId, firstName, lastName)

EXERCISE 2 (CONT'd)

Find the last name and first name of all authors who wrote books in <u>both</u> the subjects of Art and Business.

SQL



```
select lastName, firstName
from (select authorId, lastName, firstName
      from Author natural join Book
      where subject='Art'
      intersect
      select authorId, lastName, firstName
      from Author natural join Book
      where subject='Business');
```

Authors who wrote books on Art.

Select only those authors in the Art set who are also in the Business set.

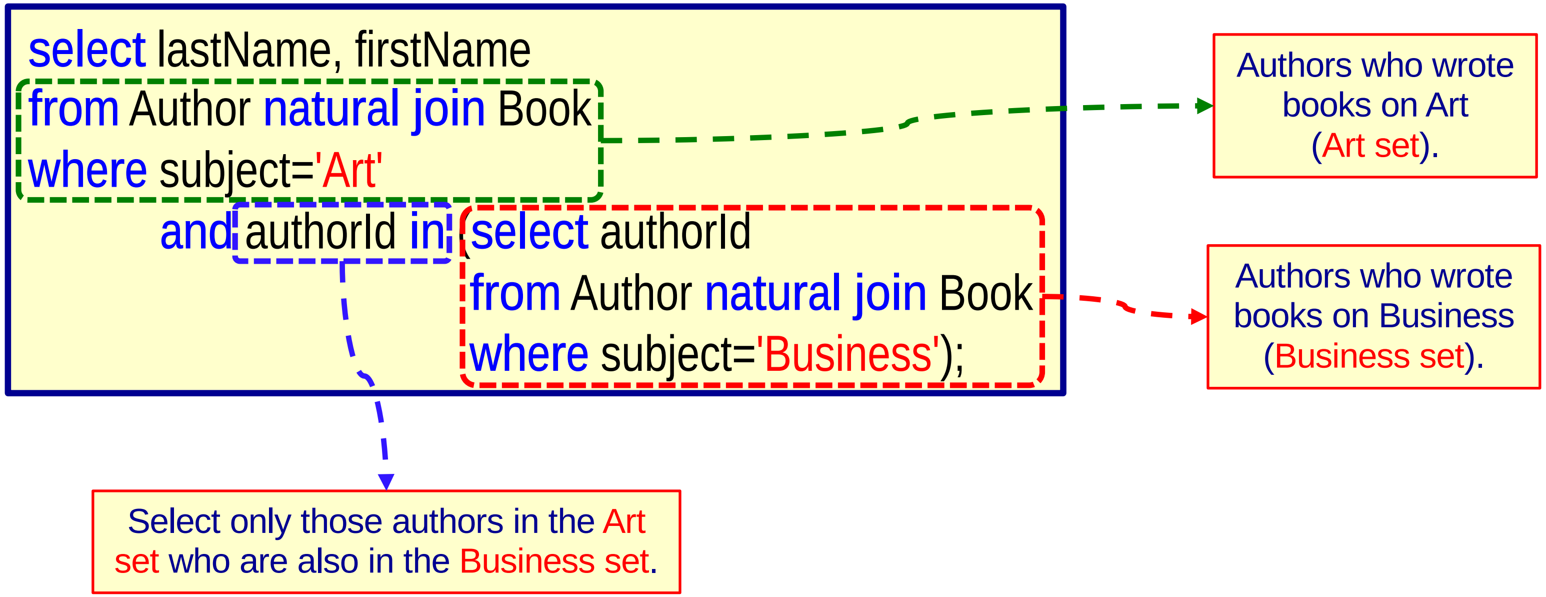
Authors who wrote books on Business.

EXERCISE 2 (CONT'd)

Use *only* set membership

Find the last name and first name of all authors who wrote books in <u>both</u> the subjects of Art and Business.

SQL



```
select lastName, firstName
from Author natural join Book
where subject='Art'
        and authorId in (select authorId
                         from Author natural join Book
                         where subject='Business');
```

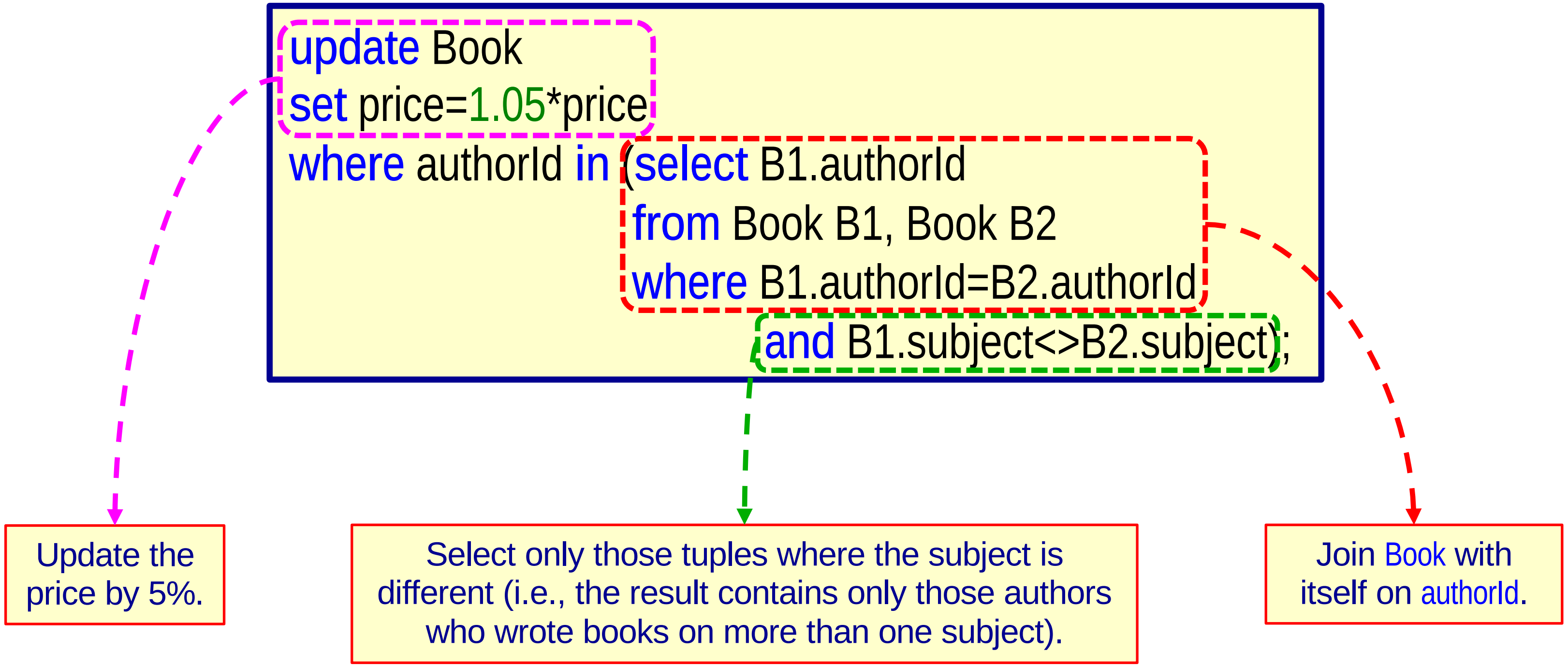
Authors who wrote books on Art (Art set).

Authors who wrote books on Business (Business set).

Select only those authors in the Art set who are also in the Business set.

EXERCISE 3

For all authors who wrote books on at least two subjects, increase the price of all their books by 5%.



```
update Book
set price=1.05*price
where authorId in (select B1.authorId
                   from Book B1, Book B2
                   where B1.authorId=B2.authorId
                         and B1.subject<>B2.subject);
```

Update the price by 5%.

Select only those tuples where the subject is different (i.e., the result contains only those authors who wrote books on more than one subject).

Join Book with itself on authorId.

Note: Natural join cannot be used if self join is required. Why?

Book(bookId, title, subject, quantityInStock, price, *authorId*)

EXERCISE 3 (CONT'D)

Do not use join

For all authors who wrote books on at least two subjects, increase the price of all their books by 5%.

```
update Book
set price=1.05*price
where authorId in (select authorId
                   from Book
                   group by authorId
                   having count(distinct subject)>=2);
```

Update the price by 5%.

Authors who wrote books on more than one subject.

Book(bookId, title, subject, quantityInStock, price, *authorId*)

Do not use subqueries.

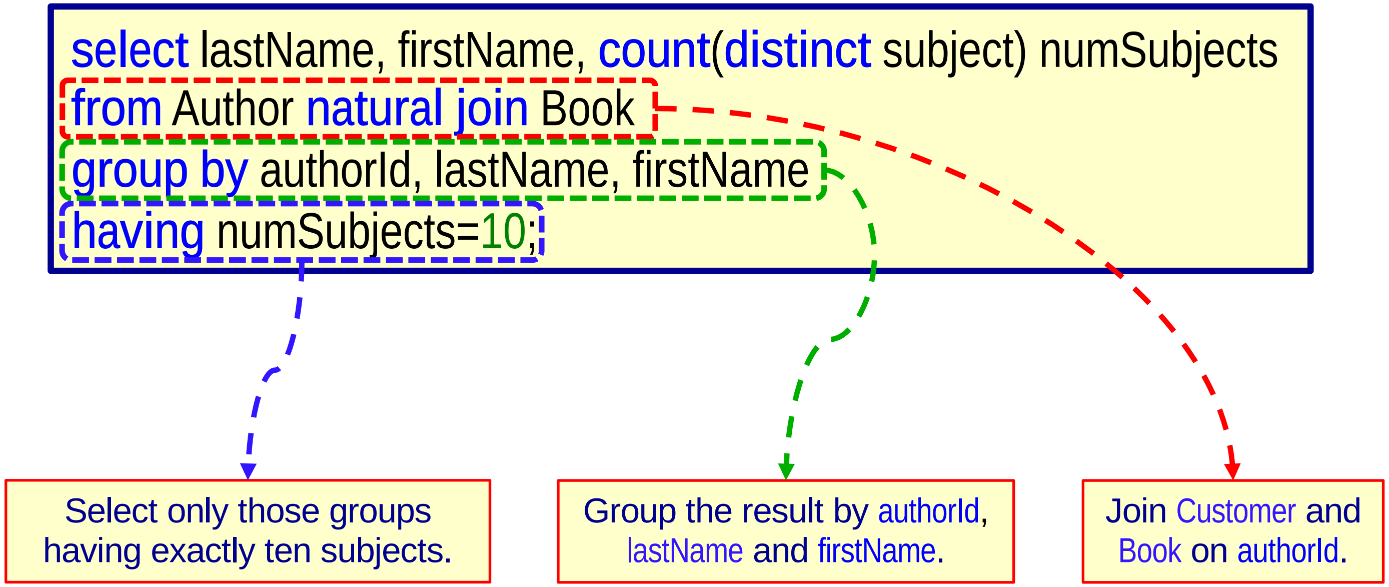
EXERCISE 4

Do not create any derived tables.

Find the last name and first name of all authors who wrote books on exactly ten different subjects.

Is this a correct solution?

NO! Why?



```
select lastName, firstName, count(distinct subject) numSubjects
from Author natural join Book
group by authorId, lastName, firstName
having numSubjects=10;
```

Select only those groups having exactly ten subjects.

Group the result by authorId, lastName and firstName.

Join Customer and Book on authorId.

☞ numSubjects **is not an attribute of either** Author **or** Book**!**

(It is an attribute only of the *final result*.)

Book(bookId, title, subject, quantityInStock, price, *authorId*)

Author(authorId, firstName, lastName)

Do not use subqueries.

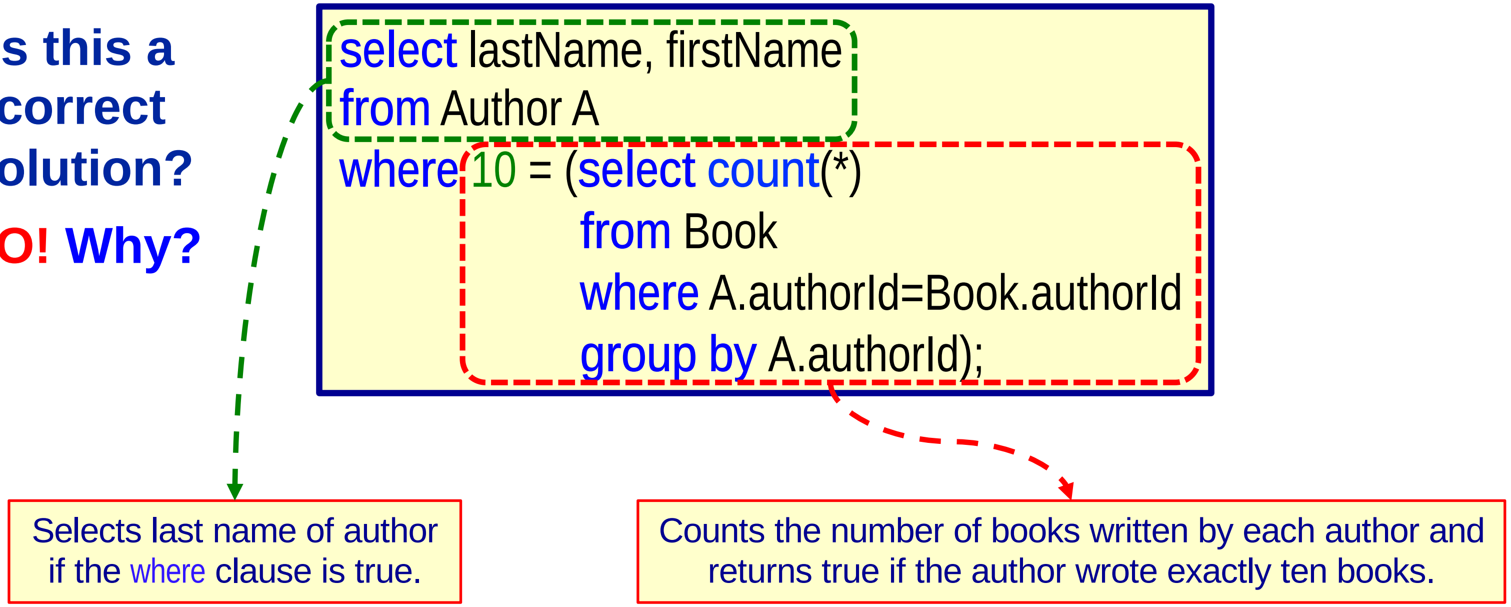
EXERCISE 4 (cont'd)

Do not create any derived tables.

Find the last name and first name of all authors who wrote books on exactly ten different subjects.

Is this a correct solution?

NO! Why?



```
select lastName, firstName
from Author A
where 10 = (select count(*)
            from Book
            where A.authorId=Book.authorId
            group by A.authorId);
```

Selects last name of author if the where clause is true.

Counts the number of books written by each author and returns true if the author wrote exactly ten books.

☞ **Selects authors who wrote exactly ten books.**

(But the subject could be the same!)

☞ **How to fix this?**

Change select count(*) **to** select count(distinct subject)**.**

Book(bookId, title, subject, quantityInStock, price, *authorId*)

Author(authorId, firstName, lastName)

Do not use subqueries.

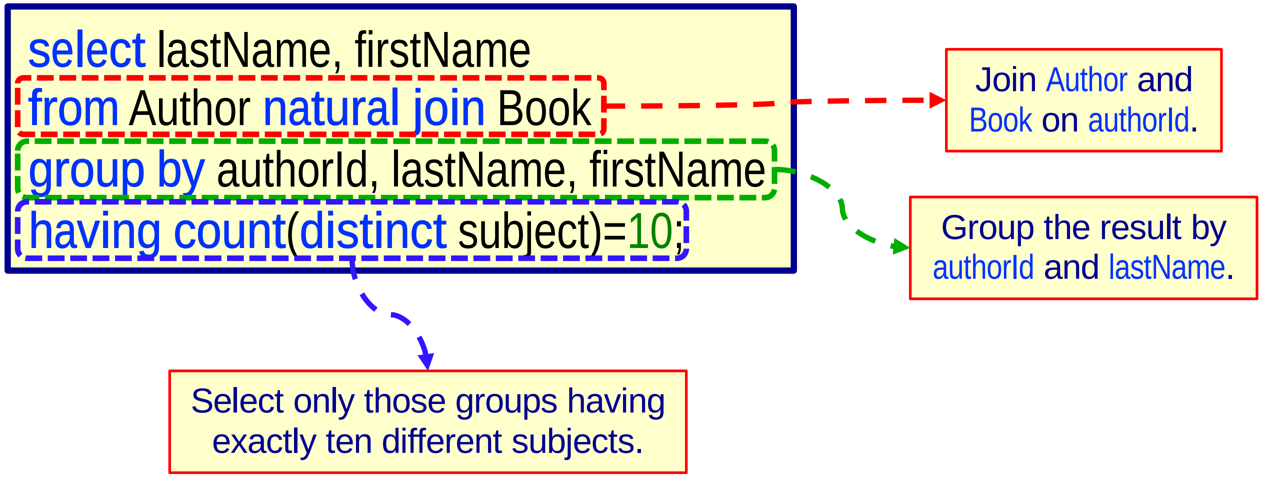
EXERCISE 4 (cont'd)

Do not create any derived tables.

Find the last name and first name of all authors who wrote books on exactly ten different subjects.

Is this a correct solution?

YES!



```
select lastName, firstName
from Author natural join Book
group by authorId, lastName, firstName
having count(distinct subject)=10;
```

Join Author and Book on authorId.

Group the result by authorId and lastName.

Select only those groups having exactly ten different subjects.

Is authorId needed in the group by clause?

☞ **YES, otherwise the count for two different authors with the same last and first name will be incorrect resulting in an incorrect result.**

Book(bookId, title, subject, quantityInStock, price, *authorId*)

Author(authorId, firstName, lastName)

Do not use subqueries.

EXERCISE 4 (cont'd)

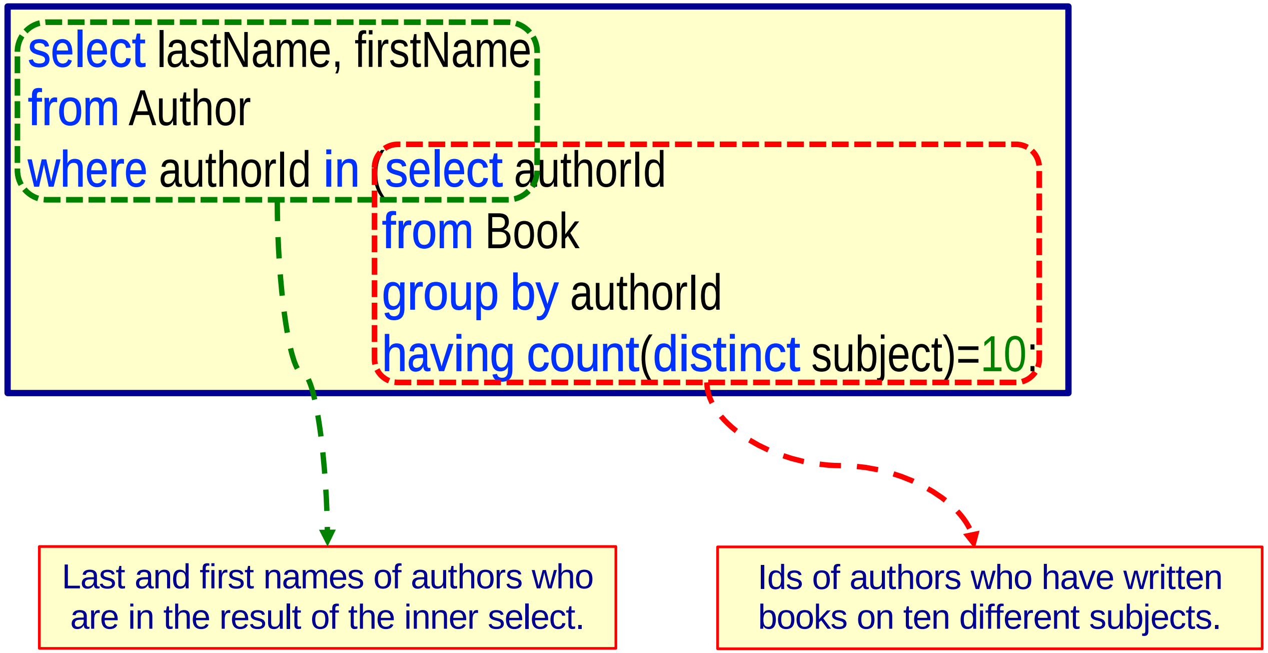
Do not create any derived tables.

Find the last name and first name of all authors who wrote books on exactly ten different subjects.

Is this a correct solution?

YES!

(But should not use subquery!)



```
select lastName, firstName
from Author
where authorId in (select authorId
                   from Book
                   group by authorId
                   having count(distinct subject)=10;
```

Last and first names of authors who are in the result of the inner select.

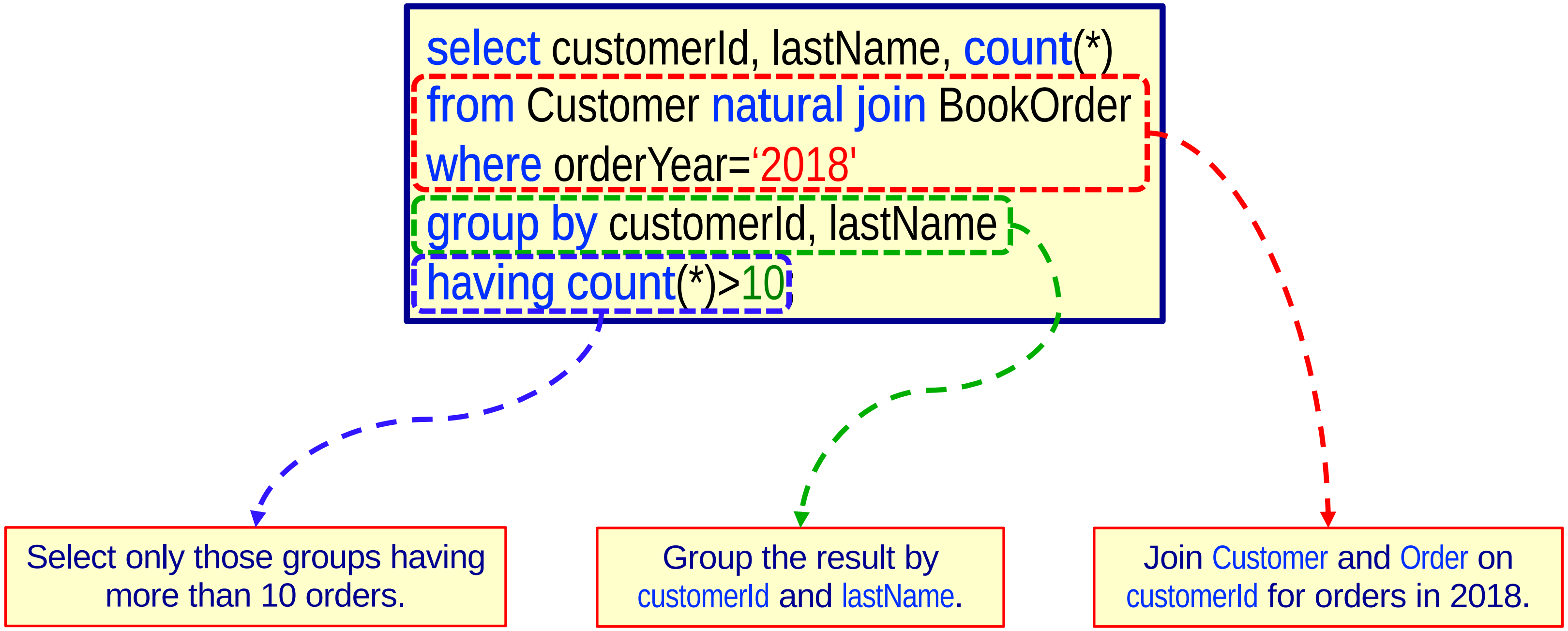
Ids of authors who have written books on ten different subjects.

Do not use subqueries.

EXERCISE 5

Do not create any derived tables.

For each customer who made more than 10 orders in 2018, find the customer id, last name and the number of orders in 2018.



```
select customerId, lastName, count(*)
from Customer natural join BookOrder
where orderYear='2018'
group by customerId, lastName
having count(*)>10
```

Select only those groups having more than 10 orders.

Group the result by customerId and lastName.

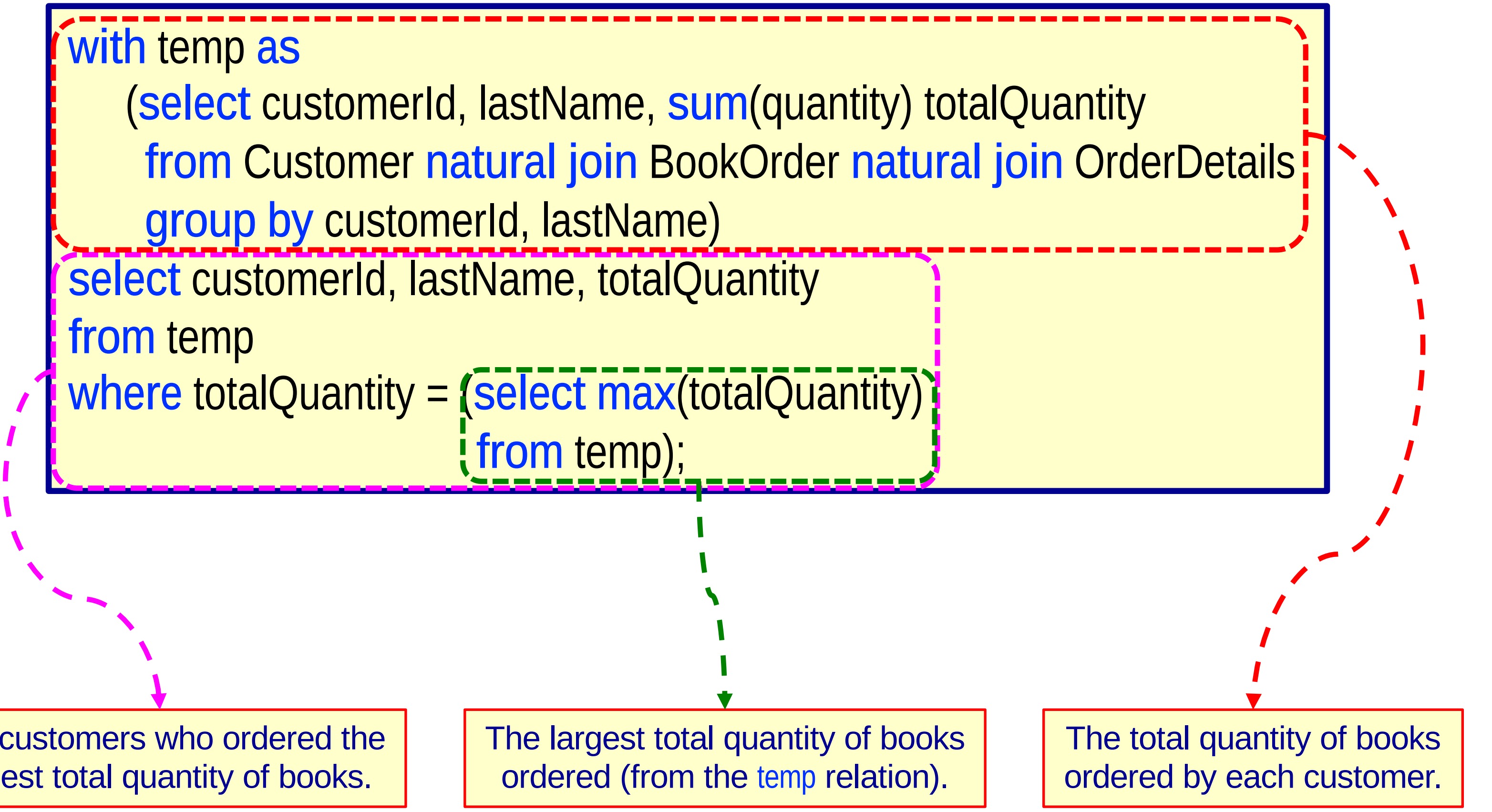
Join Customer and Order on customerId for orders in 2018.

Are both customerId and lastName needed in the group by clause?

☞ **YES, since they are both present in the select clause.**

EXERCISE 6

Find the customer id, last name and total quantity ordered for those customers who ordered the largest total quantity of books.



```
with temp as
    (select customerId, lastName, sum(quantity) totalQuantity
     from Customer natural join BookOrder natural join OrderDetails
     group by customerId, lastName)
select customerId, lastName, totalQuantity
from temp
where totalQuantity = (select max(totalQuantity)
                       from temp);
```

The customers who ordered the largest total quantity of books.

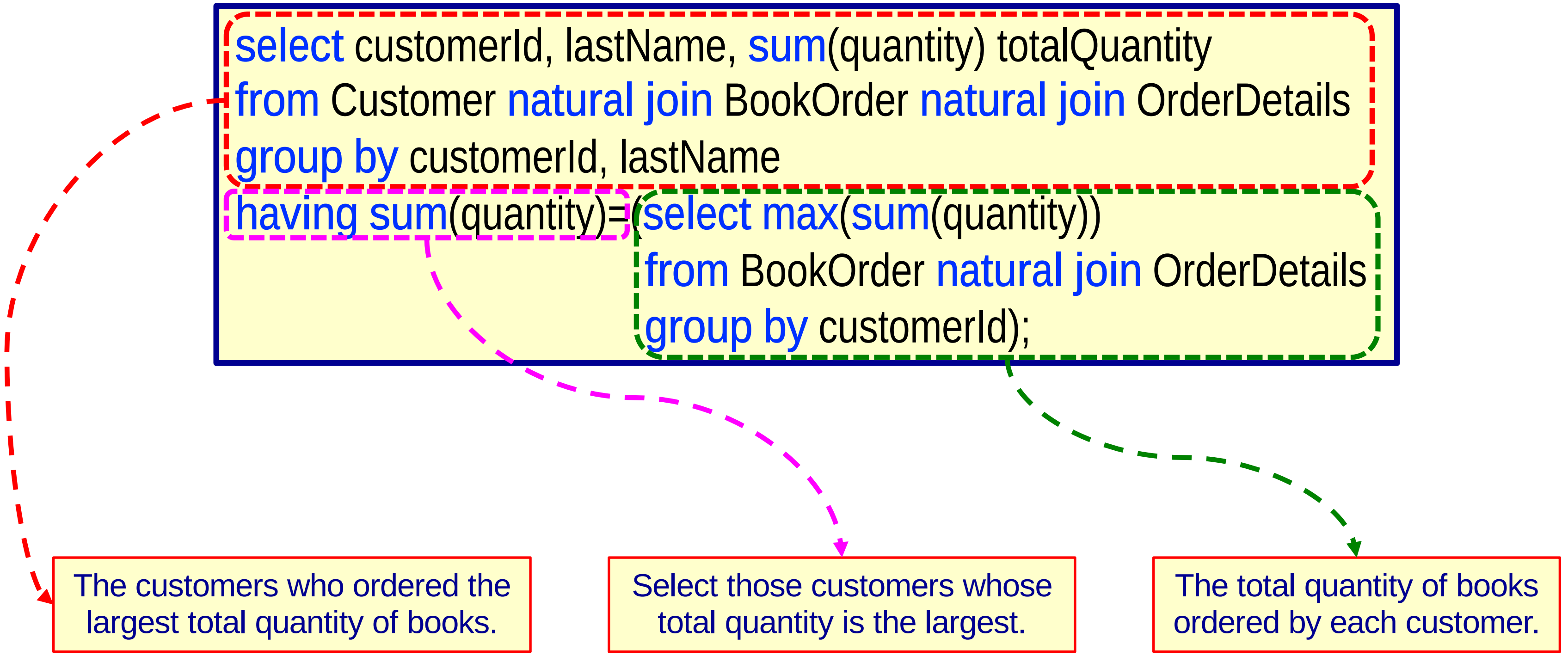
The largest total quantity of books ordered (from the temp relation).

The total quantity of books ordered by each customer.

Customer(customerId, firstName, lastName) BookOrder(orderId, *customerId*, orderYear) OrderDetails(*orderId, bookId*, quantity)

EXERCISE 6 (cont'd)

Find the customer id, last name and total quantity ordered for those customers who ordered the largest total quantity of books.



```
select customerId, lastName, sum(quantity) totalQuantity
from Customer natural join BookOrder natural join OrderDetails
group by customerId, lastName
having sum(quantity)=(select max(sum(quantity))
                      from BookOrder natural join OrderDetails
                      group by customerId);
```

The customers who ordered the largest total quantity of books.

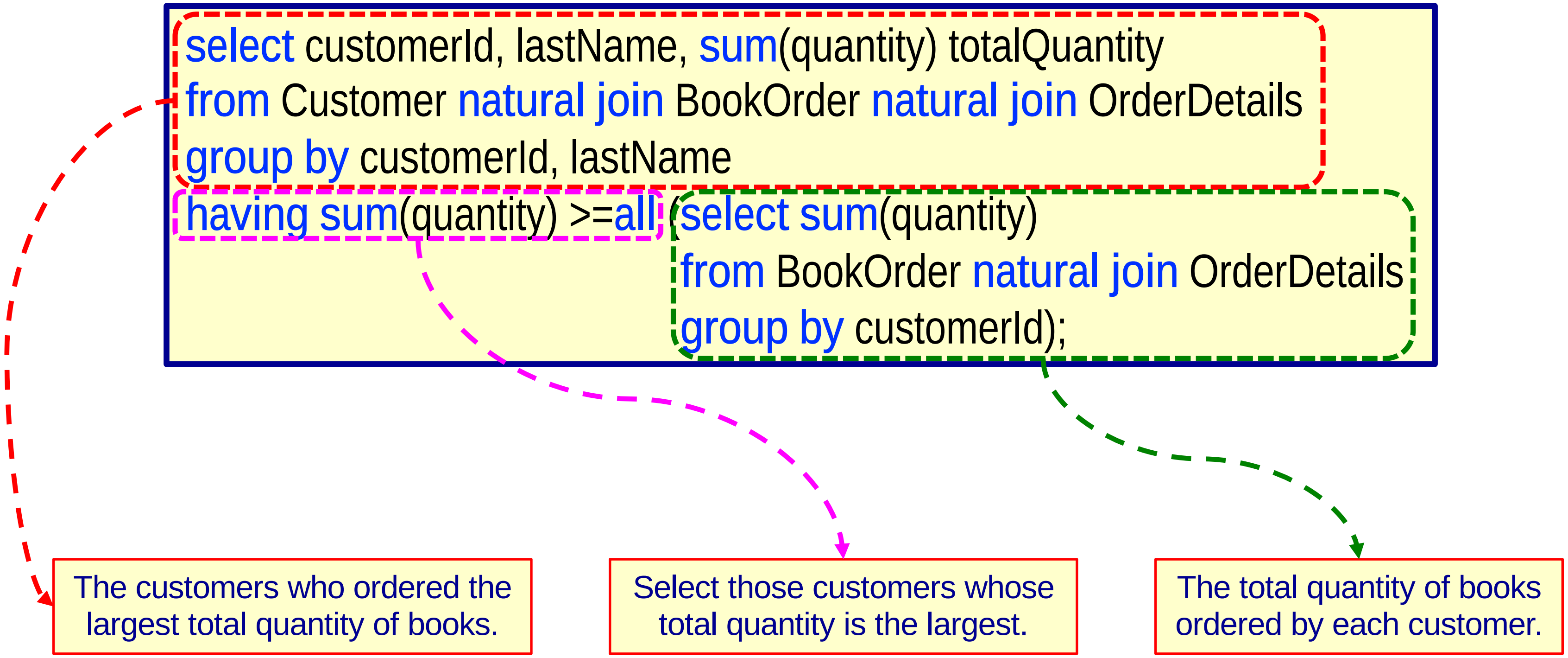
Select those customers whose total quantity is the largest.

The total quantity of books ordered by each customer.

Customer(customerId, firstName, lastName) BookOrder(orderId, *customerId*, orderYear) OrderDetails(*orderId, bookId*, quantity)

EXERCISE 6 (cont'd)

Find the customer id, last name and total quantity ordered for those customers who ordered the largest total quantity of books.



```
select customerId, lastName, sum(quantity) totalQuantity
from Customer natural join BookOrder natural join OrderDetails
group by customerId, lastName
having sum(quantity) >=all (select sum(quantity)
                            from BookOrder natural join OrderDetails
                            group by customerId);
```

The customers who ordered the largest total quantity of books.

Select those customers whose total quantity is the largest.

The total quantity of books ordered by each customer.

Customer(customerId, firstName, lastName) BookOrder(orderId, *customerId*, orderYear) OrderDetails(*orderId, bookId*, quantity)