Challeng: What’s wrong with:

SELECT

bd.author, bd.title, bd.rank,loc\_id, Count(\*) AS "count checkout"

FROM

(bookdata AS bd

INNER JOIN checkout AS c

ON bd.book\_id = c.book\_id)

INNER JOIN itemcopy i

ON bd.book\_id = i.book\_id

GROUP BY

bd.author, bd.title, bd.rank,loc\_id;

1.

SELECT \* from bookdata;

2.

SELECT \* from checkout;

3.

Select distinct book\_id from checkout;

3.a

select \* from bookdata where title = "Frank Herbert";

can anyone spot what’s wrong?

3.b

select \* from bookdata where year = 1984;

3.C

select \* from bookdata where title like '\*the\*';

3.D

select \* from bookdata where title not like '\*the\*';

3.E

select \* from bookdata where author <> "Frank Herbert";

3.F

select \* from bookdata where author = "Frank Herbert" or year = 1984;

**Exercise**:

1. Using SQL #3, What happens if you include one of the date columns in addition to the book\_id?
2. modify the bookdata query to pull: “just the facts”, author, title, and year. Go back into the SQL view, and we’re going to change the wild card \* into a list of column names.

4.

SELECT \* FROM bookdata, checkout;

5.

SELECT

\*

FROM

bookdata, checkout

WHERE

bookdata.book\_id=checkout.book\_id;

6.

SELECT

\*

FROM

bookdata bd

INNER JOIN checkout c

ON c.book\_id = bd.book\_id;

7.

SELECT

bd.rank, bd.author, bd.title, count(\*)

FROM

Bookdata bd

INNER JOIN checkout c

ON bd.book\_id = c.book\_id

GROUP BY

bd.author, bd.title;

8.

SELECT

bd.rank, bd.author, bd.title, count(\*)

FROM

Bookdata bd

INNER JOIN checkout c

ON bd.book\_id = c.book\_id

GROUP BY

bd.rank, bd.author, bd.title;

9.

SELECT

bd.rank as "Order of Goodness", bd.author as "Author", bd.title as “title”, count(\*) as "Usage Count"

FROM

Bookdata bd

INNER JOIN checkout c

ON bd.book\_id = c.book\_id

GROUP BY

bd.rank,bd.author,bd.title

ORDER BY

count(\*) DESC

Exercise

1. Convert the last SQL to a left join and then answer the following questions:
2. Are there the same number of results when it is an inner join vs an left join? What happens if you change it to a right join?
3. When you do a left join, why do some of the title have a count of 1?

10.

SELECT

bd.rank as "Order of Goodness", bd.author as "Author", bd.title as “title”, count(\*) as "Usage Count"

FROM

Bookdata bd

LEFT JOIN checkout c

ON bd.book\_id = c.book\_id

WHERE

c.book\_id is null

GROUP BY

bd.rank,bd.author,bd.title

ORDER BY

count(\*) DESC

11.

SELECT

Book\_id

FROM

Checkout c

GROUP BY

Book\_id

12.

SELECT

Rank, author, title

From

Bookdata bd

WHERE

On bd.book\_id not in (list\_of\_book\_ids\_in\_checkout);

13.

SELECT

Rank, author, title

From

Bookdata bd

WHERE

bd.book\_id not in (

SELECT

Book\_id

FROM

Checkout c

GROUP BY

Book\_id

);

14.

SELECT

bd.book\_id as id, count(\*) as c

FROM

Bookdata bd

LEFT JOIN checkout c

ON bd.book\_id = c.book\_id

GROUP BY

bd.book\_id

ORDER BY

Count(\*) DESC

15.

SELECT

bd.rank as "Order of Goodness", bd.author as "Author", bd.title as "title"

FROM

Bookdata bd

LEFT JOIN (need a tablet that has book\_ids and count of checkouts) c

ON (some criteria)

GROUP BY

bd.rank,bd.author,bd.title

ORDER BY

Bd.title DESC

15.

SELECT

bd.rank as "Order of Goodness", bd.author as "Author", bd.title as "title"

FROM

Bookdata bd

LEFT JOIN (SELECT

bd.book\_id as id, count(\*) as c

FROM

Bookdata bd

LEFT JOIN checkout c

ON bd.book\_id = c.book\_id

GROUP BY

bd.book\_id

ORDER BY

Count(\*) DESC

) c

ON bd.book\_id = c.book\_id and

c.c = 1

GROUP BY

bd.rank,bd.author,bd.title

ORDER BY

Bd.title DESC

16.

SELECT

bookdata.author, bookdata.title,

(SELECT top 1

c.startdate

FROM

checkout c

WHERE

c.book\_id = bookdata.book\_id

ORDER BY c.startdate DESC,c.id) AS MostRecentCheckOut

FROM

bookdata;

17.

SELECT

bookdata.author, bookdata.title, Max(checkout.startDate) AS MaxOfstartDate

FROM

Bookdata

INNER JOIN checkout

ON bookdata.book\_id = checkout.book\_id

GROUP BY

bookdata.author, bookdata.title;

18.

SELECT

\*

FROM

notes n

INNER JOIN notes n2

ON n.book\_id=n2.book\_id

WHERE

n.id <> n2.id

A nifty visual resource on joins: Website:

http://blog.codinghorror.com/a-visual-explanation-of-sql-joins/

Order that database server does calculations:

1. FROM clause
2. WHERE clause
3. GROUP BY clause
4. HAVING clause
5. SELECT clause
6. ORDER BY clause

Which means that calculated results are available for the order by but not the where clause. You need to use a sub query if you want to select based on the calculated results.