This will go through the tasks in A09 and at the same time review some of the material on Aggregates and Group by. Some people do understand better looking at more sql. You need to also go back to those original documents in unit 10. The focus will be on joins, aggregates, the use of distinct in the count aggregate and the Group By clause. (And also on reading the directions in the assignment and in the pdf for the AcmeBooks database.)

Task 01: Find the total amount due for all orders for book id 1108.

```
+--------+
| AmtDue |
+--------+
| 852.30 |
+--------+
```

What data do we need? The book_id for the test in the Where clause. The amount due is the product of the price per book and the quantity ordered; these are the column quantity and order_price. These columns are all in the order_details table. That defines the From clause: From bkorders.order_details .

We want to consider only book id 1108. That defines the Where clause. Where book_id in (1108).

We want to display the total amount due for this book. That defines the Select clause: Select sum(quantity * order_price) as AmtDue.

```
select sum( quantity * order_price) as AmtDue
from bkorders.order_details
where book_id in (1108);
```

We have an aggregate Sum, but we do not need a Group By since we are looking at all of the data for this particular book.

The most common problem with this task was using sum(order_price). But the assignment directions stated that the "**total amount due** … is **sum(quantity * order_price)** . That is the only correct way to calculate the total amount due for any grouping of books. This formula is used for the total sales (or orders) for a book, the total sales for a date range, the total sales for a customer, the total sales for customers living in California and the total sales for customers in California who ordered book 1162 last year.

Task 02: Find the total number of orders for book id 1108 that were placed in the previous year.

```
+----------------+
| NumberOfOrders |
+----------------+
|             21 |
+----------------+
```

What data do we need? The book_id for the test in the Where clause. We also need to test the order date. This means that we need the order_details table and the order_headers table.

We want the total number of orders. That means we need to count the orders. Orders are identified by their order_id so we should be counting the order_id.

We need two tables- is this an inner join or an outer join? Since we want orders for books, we need an inner join. So far we have

```
select ???? as "NumberOfOrders"
from bkorders.order_details  OD
join bkorders.order_headers OH  on OD.order_id = OH.order_id
where book_id in (1108)
and extract(year from order_date) =  extract(year from now())-1;
```

When we count order id over a join, we have to think about this carefully. If we use a different book_id and just look at the orders for that book, we get the following.

```
select OD.order_id, OD.order_line, OD.book_id
from bkorders.order_details  OD
```

```
join bkorders.order_headers OH  on OD.order_id = OH.order_id
where extract(year from order_date) =  extract(year from now())-1
and book_id in( 1602);
+----------+------------+---------+
| order_id | order_line | book_id |
+----------+------------+---------+
|    13145 |          1 |    1602 |
|    15025 |          1 |    1602 |
|    15025 |          3 |    1602 |
|    30855 |          1 |    1602 |
|    60039 |          3 |    1602 |
+----------+------------+---------+
5 rows in set (0.01 sec)
```

We have orders 13145, 15025, 30855, and 60039. That is 4 orders. There are 2 detail rows for book 1602 with order id 15025- but that is still one order.
We should run the query as shown here using count(distinct oh.order_id):

```
select count( DISTINCT  OH.order_id) as "NumberOfOrders"
from bkorders.order_details  OD
join bkorders.order_headers OH  on OD.order_id = OH.order_id
where book_id in (1602)
and extract(year from order_date) =  extract(year from now())-1;
```

If you omit the word Distinct then the query returns 5 and it is not counting orders- it is counting order details.

You do have to take care in deciding on count(…) and count(distinct…)

Suppose you wrote the following query: What is the difference between the two count columns?

```
select
  count( DISTINCT  OD.order_price) as "col1"
, count(OD.order_price) as "col2"
from bkorders.order_details  OD
join bkorders.order_headers OH  on OD.order_id = OH.order_id
where book_id in (1602)
and extract(year from order_date) =  extract(year from now())-1;
```

"What is the difference between Count (…) and Count (distinct …)? If the attribute value you are counting can occur more than once, do you want to count each occurrence? How is this affected by joining several tables in the From clause? Think about the difference between count(order_id) and count(distinct order_id) if you are using the order details table"

Task 03: For each customer we have in the customers table, display the customer id and last name and the number of books they have ordered and the number of books they have ordered in the current year.

```
+--------+--------------+-----------+----------------------+
| CustID | CustName     | BookTotal | CurrentYearBookTotal |
+--------+--------------+-----------+----------------------+
| 200368 | Blake        |      1110 |                  605 |
| 202958 | Denver       |      NULL |                 NULL |
| 208950 | Adams        |        20 |                   10 |
| 211483 | Carroll      |       102 |                 NULL |
| 218709 | Bonnard      |       326 |                  169 |
| 221297 | Dodgson      |        54 |                   13 |
| 222477 | Rossetti     |       145 |                   70 |
| 224038 | Austin       |       111 |                   56 |
```

Start with the first phrase in the task description: "For each customer we have in the customers table, " Some of our customers have no orders, some have one order, and some have more than one order. But we want a row for each

customer. I have 37 customers in my table, so I should get 37 rows from my query. That means we start with the customer table and do left outer joins to the other tables that we need. That will give us customers with and customers without orders, so we get all customers.
We want one row with the customer id and last name- that is easy since a customer has only one id and one last name. But we also want the number of books they have ordered- that means we need an aggregate function- sum(quantity).
Now we can think about the data values we need and the tables. We need the customer table to be certain we get all of the customers and their ID and name. Even if we were not going to display the customer name, we would still need the customer table. We need to sum(quantity) and quantity is in the order details table. And we want an aggregate based on the order_date- we need to include the order_headers table. Our From clause is

```
from bkorders.customersCS
left join bkorders.order_headers OH on CS.cust_id = OH.cust_id
left join bkorders.order_details  OD on OD.order_id = OH.order_id
```

We do not have a Where clause. We want to include all customers and their orders.

We want one row for each customer- so we need to Group by the customer id. We also want to display the customer name so our group by clause is:

```
group by CS.cust_id, CS.Cust_Name_Last
```

On to column 3: BookTotal- this one is simple; we are basing this calculation on all of the rows so this is just: sum(quantity).
Column 4 is a bit more interesting: We are including data in the expression only if the order_date is in the current year. If the order date is not in the current year, then we ignore that order data.
We have used the case expression for this type of situation:

```
case when extract(year from order_date) =  extract(year from now())-1
 then quantity end
```

If the order date is in this year, we get the quantity; if not the expression returns a null.
We can use this expression in the Sum function.

```
sum(case when extract(year from order_date) =  extract(year from now())-1
) then quantity end) as "CurrentYearBookTotal"
```

That expression can give use a null in the fourth column. You could get a 0 instead of a null by using

```
sum(case
    when year(order_date) =  year (getdate()) then quantity
    else 0
    end) as "CurrentYearBookTotal"
```

We end up with

```
select CS.cust_id as "CustID"
, CS.Cust_Name_Last as "CustName"
, sum(quantity) as "BookTotal"
, sum(case when extract(year from order_date) =  extract(year from now())-1
 then quantity end) as "CurrentYearBookTotal"
from bkorders.customers CS
left join bkorders.order_headers OH on CS.cust_id = OH.cust_id
left join bkorders.order_details  OD on OD.order_id = OH.order_id
group by CS.cust_id, CS.Cust_Name_Last
order by CS.cust_id;
```

Try the query changing these to inner joins. What is the difference?

Task 04: Display the number of orders we had in the previous quarter and the number of customers we have who had at least one order in the previous quarter. The term "previous quarter " means any date in the quarter before the current quarter. This query produces a single row with two columns.
If you run the query in Aug 2014, that is the third quarter of 2014. The query will return data for the

second quarter of 2014. If you run the query in Feb 2015, that is the first quarter of 2015, the query will return data for the fourth quarter of 2014.

This uses the idea of a previous quarter. We have seen this before with previous year and previous month. Months are cyclic; if we consider previous months we cannot do this by simply subtracting 1 from the month number. The month number for Jan 2016 is 1; the month number of the previous month is 12. Some people do this using modulo arithmetic or case statements but that is more error prone if someone else is maintaining your sql. Quarters are also cyclic: we have quarter 1, quarter 2, quarter 3, quarter 4 and then we go to quarter 1 . So you cannot just do simple arithmetic with quarters. You also need to consider the year when getting the previous quarter. The year of the previous quarter is not necessarily the current year.
I like to set this up with variables so that I can see the values I am using.

```
set @dtm_target = date_sub(current_date(),interval 1 quarter);
set @year_target = year(@dtm_target);
set @quarter_target = quarter(@dtm_target);
select @dtm_target, @year_target, @quarter_target;
```

Now to the From clause- we need the order_date and the order_id and the cust_id. These are all in the order_headers table and that is the only table we need.
We want the number of orders and the number of customers so we want the Count function in the Select clause. Now you have to decide what to count and if you need count(distinct…).
Since our only table in the From clause is bkorders.order_headers, and order_id is the primary key for that table, then each row will have a different value for order_id and we do not need to use Distinct. But a customer could have several orders in that time period and we want to know how many customers. If a customer has three orders in the previous quarter, that is still one customer. That means we need to use count(distinct cust_id).
If we had a join to the order details table in the From clause then we would need count(distinct order_id) since each order could have multiple detail rows.

```
select
   count( OH.order_id)      as "NumberOrders"
,  count(distinct cust_id) as "NumberCustWithOrders"
from bkorders.order_headers OH
where  year(order_date) = @year_target
and quarter(order_date) = @quarter_target
;
```

Task 05: Which book with the category Data Storage Techniques has the most orders? We are using the number of orders for the book - not the quantity. Display the book id and title. Consider there might be ties for first place- in that case all tied books should be returned.

This task asks about books in a specific category. "Tasks may ask you to filter for book topics in defined sets of values called **categories**; use the definitions of the categories in the pdf file for the books tables. A book topic is not the same thing as a book category." Making up your own categories is not accepted. (Remember the list of reptiles and rodents- same issue here.)

What data do we need? We need the book_id and title (in the books table). We want to count the number of orders so we need the order_id which is in the order_details table. We need to filter on the topic id which is in the book_topics table. These three tables will form a join. We want inner joins since we want books with orders and with topics.

```
from bkinfo.books  BK
join bkinfo.book_topics BT on BK.book_id = BT.book_id
join bkorders.order_details OD on BK.book_id = OD.book_id
```

We want to filter by the topic ID for the indicated category:

```
WHERE BT.topic_id IN ('NOSQL', 'XML', 'DB')
```

And since we want data by book we need to group by the book_id and also by the title so we can display the title. With MySQL 5.6 you can use group by bk.book_id)

```
group by  BK.book_id, BK.title
```

The main query gathers the data for books and the number of orders for each book. The count(distinct order_id) test is placed in a having clause. We use a similar query as a subquery in a From clause which select the values for count(distinct order_id) for each book. The rest of that subquery uses max(TopicOrders.BookOrderCount) to get the largest value for the bookOrderCount and that is used to compare to the test in the main query.
This is a fairly complex query. The inner most subquery gets the number of orders per book; the next level going up gets the largest of those values; the top level recalculates the number of orders per book and filters for that value being equal to the largest value.

```
select BK.book_id, title
from bkinfo.books  BK
join bkinfo.book_topics BT on BK.book_id = BT.book_id
join bkorders.order_details OD on BK.book_id = OD.book_id
where BT.topic_id in ('NOSQL', 'XML', 'DB')
group by BK.book_id, title
having count(distinct order_id) = (
    select max(TopicOrders.BookOrderCount)
    from(
        select count(distinct order_id) as BookOrderCount
        from bkinfo.books  BK
        join bkinfo.book_topics BT on BK.book_id = BT.book_id
        join bkorders.order_details OD on BK.book_id = OD.book_id
        where BT.topic_id in ('NOSQL', 'XML', 'DB')
        group by BK.book_id
            ) TopicOrders
      );
```

If two or more books have the same value for BookOrderCount, they will all be returned.

Task 06: Use the cross tab techniques described in the notes for this task.
We want to know how many books we have in the books table in each of the indicated topic **categories**.
Display the result as a single output row. The categories to use are:

- Science
- Database Systems
- Data Storage Techniques

There is also a column for all books.
If a book is in more than one category, then it counts in each of those categories.

```
Science          Database Systems Data Storage Techniques  All Books
---------------- ---------------- ----------------------- ----------
               0              456                      73       1295
```

This task also uses book categories. "Tasks may ask you to filter for book topics in defined sets of values called **categories**; use the definitions of the categories in the pdf file for the books tables.
Since we want to know how many books are in each category we need the count function with the book_id

We have some books with no topic, some with one topic and some with more than one topic.
Since the last column is the count for all books, we need to use the books table left join the book topics table.

```
from  bkinfo.books BK
left join bkinfo.book_topics BT on BK.book_id = BT.book_id
;
```

Now on to the Count function. If the book has a topic of SCI, it is counted in the Science category. We can use the same technique for this as we did in Task 03 when calculating data for books ordered in the current year- a case expression: case when topic_id in ('SCI') then BK.book_id else null end.

The other columns are similar: case when topic_id in ('DB', 'SQL',SSRV', "MySQL', 'ORA', 'ADO') then BK.book_id else null end).

Is this a count(. . .) or a count(distinct …)?
We have a book id 1105 SQL:1999 Relational Language Concepts with a topic of DB and a topic of SQL.Is this one book or two? There is only one book id value for this book- so this is a single book and we need to use count(distinct…). What we are counting is the value returned by the case expression which is either book_id or null.

```
count(distinct case when topic_id in ('DB', 'SQL', 'SSRV', 'MYSQL', 'ORA', 'ADO')
then BK.book_id else null end) as "Database Systems"
```

Should we use count(distinct …) for the Science category? We don't really need it here since there is only one topic id but you can use distinct to make this query more uniform.. The last column for all books needs count(distinct book_id).
You can do a simple Select count(*) from bkinfo.books BK to check the value for the last column.

What about a book with two topic id from different categories? Book 1483 Programming with XML is listed as both an XML book and an SQL book so it fits into two categories. You can add a Where clause to the above query: where Bk.book_id = 1483 to check that the query now return a 1 for "Database Systems" and for "Data Storage Techniques" as well as for "All Books"

```
select
   count(distinct case when topic_id in ('SCI') then BK.book_id else null end) as "Science"
,  count(distinct case when topic_id in ('DB', 'SQL', 'SSRV', 'MYSQL', 'ORA', 'ADO') then
BK.book_id else null end) as "Database Systems"
,  count(distinct case when topic_id in ('NOSQL', 'XML', 'DB') then BK.book_id else null
end) as "Data Storage Techniques"
,  count(distinct BK.book_id)  as "All Books"
from  bkinfo.books BK
left join bkinfo.book_topics BT on BK.book_id = BT.book_id
;
```

Task 07: For each customer that we have in the customer table, list the customer id and last name and the date of the most recent order for that customer.
If the customer has no orders, then display "No Orders' in the column for the order date.
For this query, if there are order headers that have no associated order detail rows, those orders should not be included.

```
+--------+--------------+-----------------+
| CustID | CustName     | MostRecentOrder |
+--------+--------------+-----------------+
| 200368 | Blake        | 2016-07-28      |
| 212921 | Books on Tap | 2016-03-22      |
| 217002 | Grieg        | No Orders       |
| 217796 | Anders       | 2016-05-06      |
| 221297 | Dodgson      | 2016-05-06      |
```

The tasks description starts: " For each customer that we have in the customer table," so we are going to need customer left join . . .
But we want to include only orders headers that have associated order detail rows: that needs an inner join between these two tables. Your query has to refer to the OD tables to meet the task requirements.
One way to handle this is as shown here. We do the inner join first getting a temporary table of orders headers that have order details. Then we do a Right join to customers. That gives use OH with OD rows with associated customers and also customers with no associated orders.

```
from  bkorders.order_details  OD
join  bkorders.order_headers OH on OH.order_id = OD.order_id
right join bkorders.customersCS on OH.cust_id = CS.cust_id
```

We want one row per customer so we need to group by the customer id and since we want to display the customer name, we add cust_name_last to the Group by clause. With MySQL 5.6 you can use group by CS.cust_id

We want the most recent order date for each customer. That is Max(order_date). But it is possible that a customer has no orders and that expression would be a null. When we want a null column to display a different value, we can use coalesce so we have coalesce(max(order_date), 'No Orders') as "MostRecentOrder".

```
select CS.cust_id as "CustID"
, CS.Cust_Name_Last  as "CustName"
,  coalesce(max(order_date) , 'No Orders') as "MostRecentOrder"
from  bkorders.order_details OD
join  bkorders.order_headers OH on OH.order_id = OD.order_id
right join bkorders.customers CS on OH.cust_id = CS.cust_id
group by CS.cust_id, CS.Cust_Name_Last
order by CS.cust_id;
```

Task 08: Display the customer id and last name for customers who have ordered more than $500 worth of books with a topic of SQL but who have never order book id 1142.

This is a type of query you have seen before and many people still get wrong. We want to display customer data for customers who meet two tests.
1) they have ordered more than $500 worth of books with a topic of SQL
and
2) they have never ordered book id 1142
The outline for this query can be done with two subqueries.

```
select cust_id, cust_name_last
from bk_customers
where cust_id in (
list of customers who meet test 1: more than $500 worth of books with a topic
of SQL
)
and cust_id NOT IN (
list of customers who bought book 1142
)
```

This idea will keep coming up over the semester (as on the final exam and more assignments). There is a difference between a customer who has not ordered book 1142 and a customer who has ordered a book that is not book 1142. These are logically different questions and are coded in different ways.
The second subquery needs the OH table and the OD table so that it can compare cust_id values and test on the book_id.

```
and cust_id NOT IN (
  select cust_id
  from bkorders.order_headers OH
  join bkorders.order_details  OD on OD.order_id  = OH.order_id
  where book_id = 1142
```

The first subquery uses aggregate data : sum(quantity* order_price) compared to $500. You are not allowed to use an aggregate in a Where clause, so this needs to be in a Having clause.

```
  having sum(quantity* order_price) > 500
```

This subquery needs the OH table to get the cust_id for comparison; it needs the OD table for the quantity and order_price; it need the book_topics table to check for topic_id = 'SQL'
Since we have an aggregate, we need to group by the cust_id so we get the sum for each customer separately.
The first subquery is

```
where cust_id in (
  select cust_id
  from bkorders.order_headers OH
  join bkorders.order_details  OD on OH.order_id = OD.order_id
  join bkinfo.book_topics BT on BT.book_id = OD.book_id
  where topic_id = 'SQL'
  group by cust_id
  having sum(quantity* order_price) > 500
```

General:
This assignment- as is true for all assignments, relies on your remembering the material in the earlier units. If you had troubles with the difference between inner and outer joins and decided to forget about outer joins- that will not work. Three of these tasks required outer joins. You have to deal with date issues and date functions- you might not remember all of the syntax for these functions but you should be able to look them up and find the correct function. You need to remember that some date components are cyclical in nature and why that matters. And you need to remember those multi-match/non match query patterns.
A surprising number of people were using count(*) That is useful only for counting the number of rows in a table. You lose all control over what specific data you are supposed to be counting and have no way to handle count(Distinct …)

For this unit's material:

You have to decide on the proper Grouping columns- if we want data related to each customer you will need to Group By cust_id. If you group by the Cust_id and Order_id, you get finer/smaller groups based on the order_id.

If you need to find the number of occurrences of something, you use Count; if you need the total of something, you will use Sum. If you need the biggest or smallest of something you use Max and Min. These aggreagate function take column names, or expression based on columns as arguments.

If you need to count the number of order id values, count the order_id column. If you need the total quantity of books ordered, sum the quantity column. If you need the total amount due, sum(quantity * order_price).

And it also helps if you read the task descriptions reading for details.

.