Due Date: Saturday, November 02, 2013 11:00 PM
Points: 40 points max
Turn In: The script and spool files turned in via the assignment drop box

General Directions

Use the books databases.

These tasks focus on the use of Subqueries. Consequently, you **must use subqueries** to solve the problems. In many cases you could solve the task without the use of subqueries- **but that will not earn any credit** for the assignment. Many of these will use aggregate functions.

- You must do each task with a single query. The query will have sub components but it will be a single query.
- Use only the data supplied in the task to write the query and do not make assumption about the data that are not supported by the create table statements. For example, Task 01 asks you to find customers who have bought any book by the author with the last name of "Celko". You use the literal “Celko” in your query. You are not allowed to manually read the tables and find the author id for Celko; you are not allowed to assume there is only one author with that last name.
- Do not use a variable.
- **Do not use a join of any kind and do not use a correlated subquery. Do not use a comma join. If you use a join, you will get no points for that task. A From clause will reference only one table**
- Do not use set operations (Union)
- If you use column aliases, then use different column aliases in each subquery. Using the same column alias may be legit but it makes your query very hard to read.
- A book with an order quantity of 0 is still considered a book that is ordered. (Some people are testing that the quantity >0- do **not** do that to determine if a book has been ordered.). An order header without any detail lines is still an order.
- Queries that use sub queries tend to be longer and harder to read if they are not formatted properly. **Queries that are hard to read will lose points.** The key words From, Where, Group By, Order By, Having start new lines and need to align. The indentation should be 3 or 4 spaces.
- Several tasks ask you to display a "winner" such as the customer(s) with the most orders. The result set might contain one row if there is a single customer with the most orders, or possibly more than one row if more than one customer have the same highest number for orders. This does **not** mean to display all customers in some order- display only the winners.

Acceptable alignment for a subquery. Note that you can easily find the subquery.

```
select cust_id,  ord_id
,  (
      select sum(quantity_ordered)
      from a_oe.order_details  OD
      where OH.ord_id = OD.ord_id
   ) as "NumItemsPerOrder"
from a_oe.order_headers  OH;
```

Not acceptable- the subquery runs on a single line making it harder to see the components

```
select cust_id,  ord_id,
( select sum(quantity_ordered) from a_oe.order_details OD where OH.ord_id = OD.ord_id) as
"NumItemsPerOrder"
from  a_oe.order_headers  OH;
```

Not acceptable: The Select for the subquery needs to be indented a few spaces. It is hard to see where the the subquery starts and ends.

```
select cust_id,  ord_id,  (
select sum(quantity_ordered)
from a_oe.order_details  OD
where OH.ord_id = OD.ord_id) as "NumItemsPerOrder"
from a_oe.order_headers  OH;
```

Not acceptable: The Select for the subquery needs to be indented a few spaces- not a lot of spaces. This also is hard to read.

```
select cust_id,  ord_id,  (
                           select sum(quantity_ordered)
                           from a_oe.order_details  OD
                           where OH.ord_id = OD.ord_id) as "NumItemsPerOrder"
from a_oe.order_headers  OH;
```

Tasks

Task 01: Display the ID and last name of the customers who have bought any book by the author with the last name of "Celko". (Think about how many tables this needs- start from the inner most subquery (the one that filters for Celko)- and work out.)

Task 02: Display the book id and title for any books which someone has ordered and the book is **both** an SQL book and a database book. Use the Topic_id to filter for DB and SQL. Sort by the book_id.

Task 03: Display the book id and title for any books which someone has ordered and the same book has a topic of DB but does not have a topic of SQL. Use the Topic_id to filter for DB and SQL. Sort by the book_id.

Task 04: Display the ID and title of the book(s) with the largest number of sales; include ties. For this query, use the total quantity sold when determining the sales of a book.

Task 05: Display the ID and title of the book(s) with the largest sales amount; include ties. For this query, use the total extended cost when determining the sales of a book.

Task 06: Display the ID and last name of the customer(s) with the most orders; include ties. The phrase "most orders" refers to the number of orders. Use the ALL operator in this query.

Task 07: Display the customer id and last name for customers with no more than 5 orders. Do not include any customer with no orders. Sort by the customer id.

Task 08: For each book in the books table that includes 'Bird 'in the book title, display the book ID and title and a message as to whether or not we have any orders for that book. Do not use the Count() function.
The output will follow this format and be sorted by the OrderStatus column with the books with no order first; the second sort key is the book id.

```
Sample rows only
+---------+-------------------------------------------------+-------------+
| book_id | title                                           | OrderStatus |
+---------+-------------------------------------------------+-------------+
|    2029 | The Forgotten Bird Strikes Back                 | No Orders   |
|    1104 | Sibley Guide to Bird Life and Behavior          | No Orders   |
|    1543 | Birding and Implementing a Birdhouse            | Have orders |
|    1085 | My Bird Flicker                                 | Have orders |
```