In-Class Practice

Steps for building complex queries

1. State the problem to be solved in English.
2. Use pseudocode to outline the query. Start by figuring what you need to find first and make that your inner query.
3. If necessary, use pseudocode to outline each subquery.
4. Code the subqueries and test them individually
5. Combine inner queries with outer queries and test the final query.

WHERE clause subquery

1. Write a query that selects the AVG invoice_total from the invoice table and then uses that query as a subquery to a query that pulls all invoices that have totals greater than the average invoice total. *i.e. A single value in a subquery.* **Discuss the following with your partner:** *Do you think you could do this without a subquery?*

2. Pull invoices (specifically the invoice_id, invoice date, invoice total) from all vendors in TX, TN, and NV. *i.e. Multiple values in the subquery.* **Discuss the following with your partner:** *Could you figure out how to do this without a subquery? Try!*

3. Update the query to pull all invoices except for vendors in TX, TN, and NV. What change did you have to make to do this? **Discuss it with your partner.**

4. Pull all invoices (all columns) that have an invoice_total greater than **all** of the individual invoice_totals of the vendor with the name 'IBM'. *Hint: Talk with your partner about what data you need to find in your first subquery. Use this discussion to take note on how to utilize the ALL keyword.*

5. Update query to pull the invoices that are greater than just **some** of IBM's invoices

FROM clause using subquery like a table

6. Find the average count of invoices for all vendors that have sent more than 1 invoice but while ignoring vendor 123 who has sent us more than 40 invoices. *Hint: Start by finding the count of invoices for each vendor and filtering out the ones that have a count of 1. Then use this query as a subquery but selecting the average count from it. After you complete this add in a filter to ignore vendor_id 123 and you should see the average go down since they had a lot of invoices.*

7. Combine the two queries below like they are tables using an in-line join in the FROM clause. Join them like you would two tables where both the vendor_state and invoice_totals match and then select vendor_name, vendor_state, and invoice_total. *Hint: Start with Select*

```
--FIRST QUERY...Give this a table alias of summary_1
SELECT  v.vendor_id,
        v.vendor_name,
        v.vendor_state,
        SUM(i.invoice_total) AS INVOICE_TOTAL
FROM invoices i JOIN vendors v ON i.vendor_id = v.vendor_id
GROUP BY v.vendor_id, v.vendor_name, v.vendor_state
order by vendor_state;

--SECOND QUERY....Give this a table alias of summary_2
select vendor_state, max(sum_of_invoices) as invoice_total
from       (SELECT v.vendor_id,
             v.vendor_state,
               SUM(i.invoice_total) AS sum_of_invoices
        FROM invoices i JOIN vendors v ON i.vendor_id = v.vendor_id
        GROUP BY v.vendor_id, v.vendor_state
        ORDER BY v.vendor_state)
GROUP BY vendor_state;
```

MORE PRACTICE YAY!!!!!!

8. Using a subquery, tell me how many employees in each department are currently assigned to projects.
9. Using one or more subqueries, give me a list of full customer name, customer city, and customer state from the customers_ex table where the customer is the only customer living in their state.
10. Write a query, using subqueries, to show the columns customer full name, customer state, number of orders for the customer, and average processing time (shipped_date – order_date) for each customer, for all customers whose average processing time is greater than the average processing time for all customers.

If you get done and need more to do, start by describing the differences in using a subquery in the WHERE clause and the FROM clause.

Which one is better for which situations?

WHERE better for ____

FROM better for ____