# Lab 3 - Chapter 5 -Table Expressions

## Derived Tables, Common Table Expressions, & Views

**Code the Following questions using the Antiques Dataset**

**Please read the section on derived tables and CTE very carefully before beginning this lab. Table Expressions require strict adherence to rules (i.e., alises). You can also use the class slides to help guide you through many of the problems.**

*For full credit, submit both your **coding answers** along with a **snipped image of your code output that adheres to screenshot requirements**. Both of the below examples qualify for the requirements. Also, some questions require explanations, in addition to code and code output.

An example is provided below:

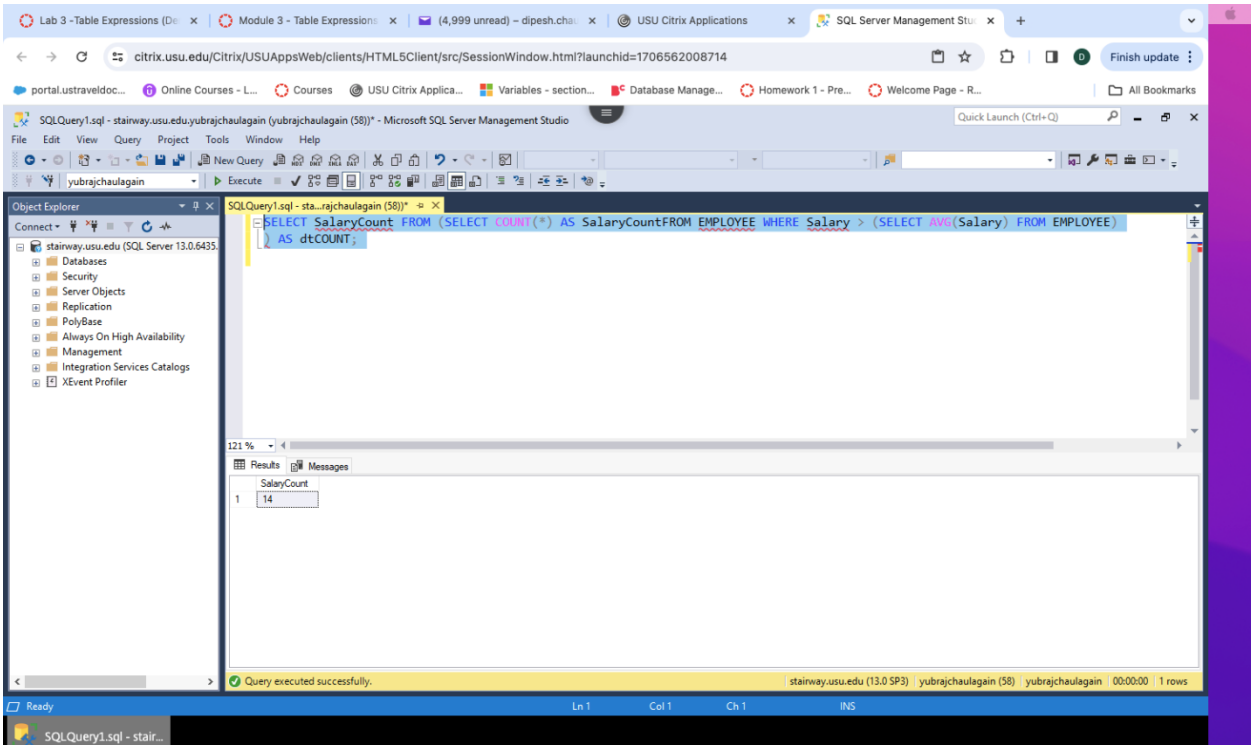
On a PC, you can use the Snipping tool, and on MAC, Command/SHIFT/4 allows you to snip. Please don't take entire screen captures. Google can provide additional information for both snipping and the MAC capture.

1. Write a Derived Table called dtCOUNT that counts the employees making over the average salary. Name your COUNT Alias SalaryCount. In addition to the Derived Table, you need to include a Type 1 Subquery. Your SELECT statement should read: SELECT SalaryCount

***If you need help with the Subquery to be placed in the Derived Table, see the solution at the bottom of this lab.

SELECT SalaryCount FROM (SELECT COUNT(*) AS SalaryCountFROM EMPLOYEE WHERE Salary > (SELECT AVG(Salary) FROM EMPLOYEE)

) AS dtCOUNT;




2. Why were you able to refer to an Alias in the SELECT statement?

Because of the logical order of processing the query.

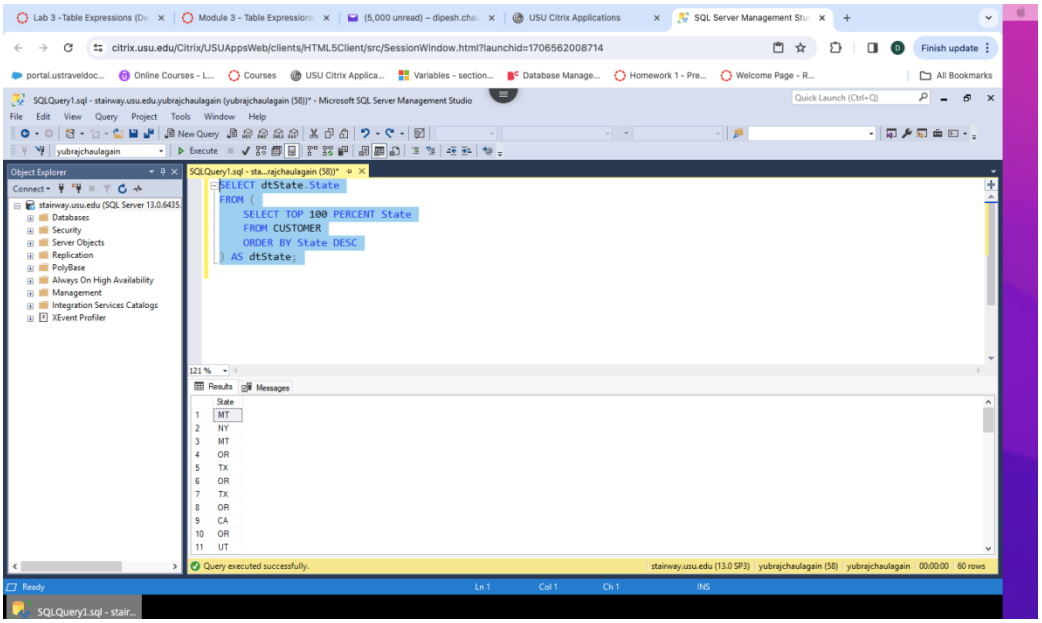
3. Run the following code and describe the error message you received:

The ORDER BY clause is invalid in views, inline functions, derived tables, subqueries, and common table expressions, unless TOP, OFFSET or FOR XML is also specified.

```
SELECT dtstate.State
FROM (SELECT State
        FROM CUSTOMER
      ORDER BY State DESC) as dtState;
```

4. Rewrite the code above and place the ORDER BY clause in a location that will allow the code to run without an error message.

```
SELECT dtState.State
FROM (
   SELECT TOP 100 PERCENT State
   FROM CUSTOMER
   ORDER BY State DESC
) AS dtState;
```




5. Examine the following code example and indicate if it will run or not. If not, what is causing the code to fail?

Incorrect syntax due to not having the alias as dt state After closing ( SELECT state  FROM Customer);

SELECT dtstate.State

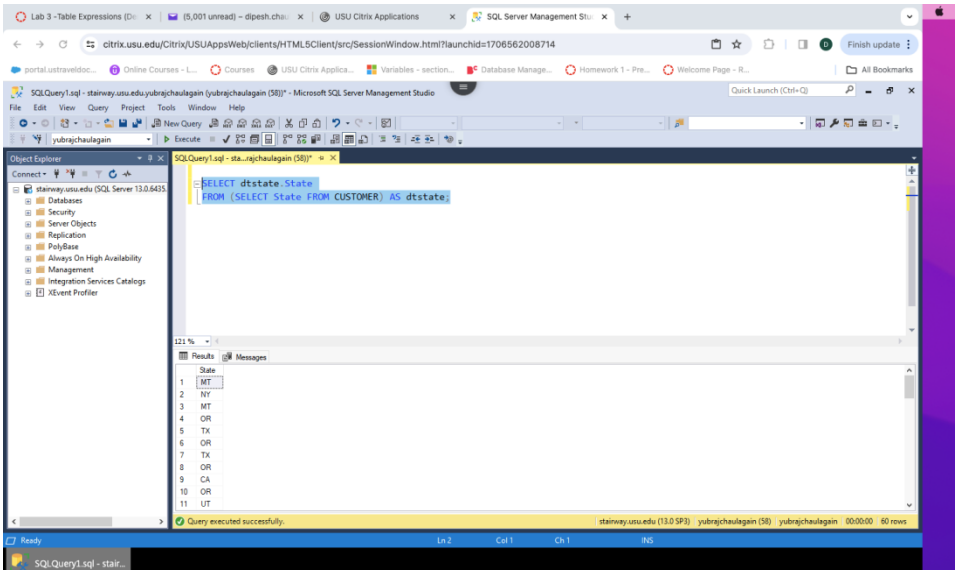
FROM (SELECT State

FROM CUSTOMER);

The correct version of the code is

SELECT dtstate.State

FROM (SELECT State FROM CUSTOMER) AS dtstate;




6. Why doesn't the following Derived Table run? Based on your explanation, fix the code so it will run.

SELECT e.LastName

FROM

(SELECT LastName FROM Employee WHERE Gender = 'Male' ) AS e JOIN

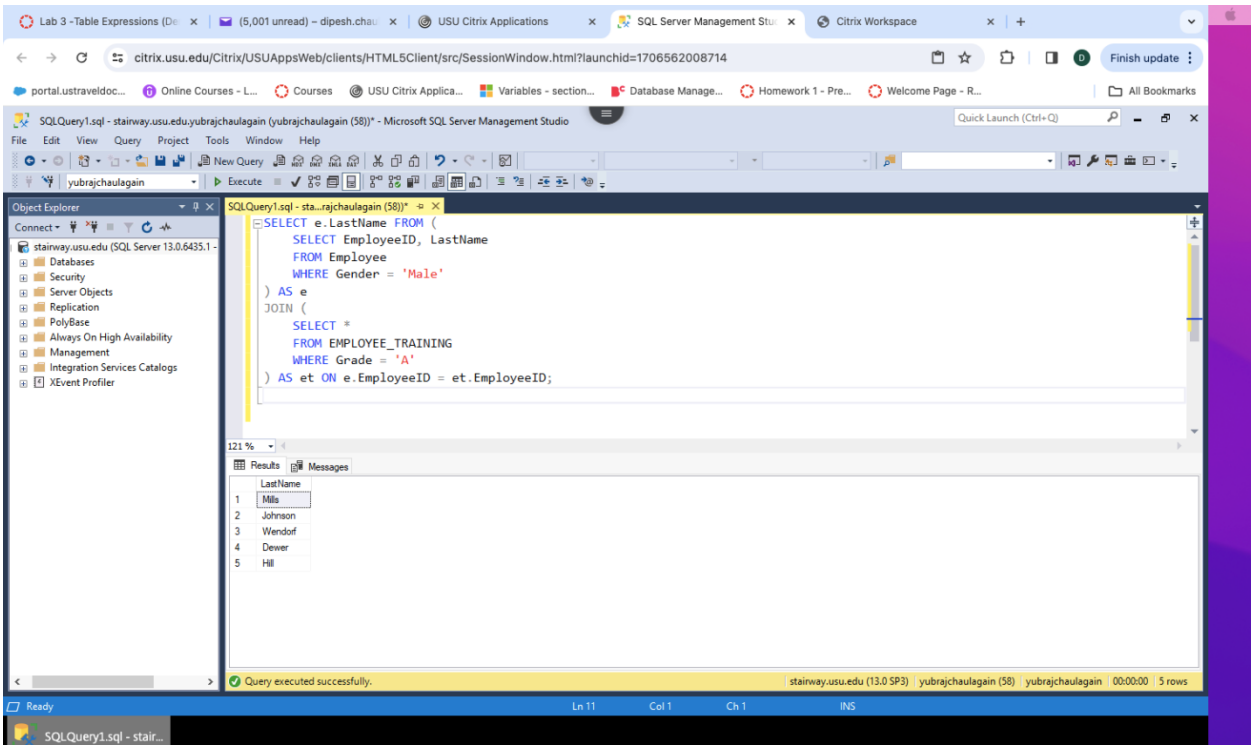
(SELECT * FROM EMPLOYEE_TRAINING WHERE Grade = 'A') AS et

ON (e.EmployeeID = et.EmployeeID);

The query seems to be aiming to select the last names of male employees (Employee table) who have a grade of 'A' in the EMPLOYEE_TRAINING table.

The issue with the query is that the derived table e is defined with an alias (AS e), but it's missing a column name after the AS. In a derived table, you need to specify the columns that you're selecting.

```
SELECT e.LastName FROM (
    SELECT EmployeeID, LastName
    FROM Employee
    WHERE Gender = 'Male'
) AS e
JOIN (
    SELECT *
    FROM EMPLOYEE_TRAINING
    WHERE Grade = 'A'
) AS et ON e.EmployeeID = et.EmployeeID;
```

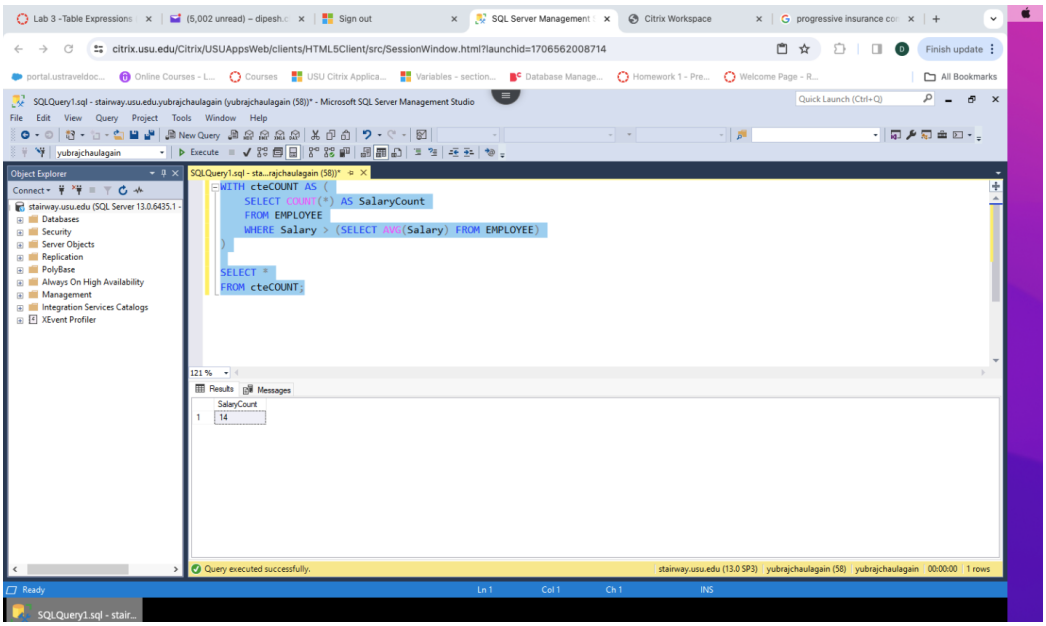



7. Write a Common Table Expression called cteCOUNT that counts the employees making over the average salary. Name your COUNT Alias SalaryCount. As part of the CTE, you need to include a Type 1 Subquery. *Run SELECT * FROM cteCOUNT to see if your CTE runs.

```
WITH cteCOUNT AS (

    SELECT COUNT(*) AS SalaryCount

    FROM EMPLOYEE

    WHERE Salary > (SELECT AVG(Salary) FROM EMPLOYEE)

)
```

SELECT *

FROM cteCOUNT;




8. If you change SELECT * FROM cteCount to SELECT SalaryCount FROM cteCount, does the code still run?

Yes, it still run with the same output.

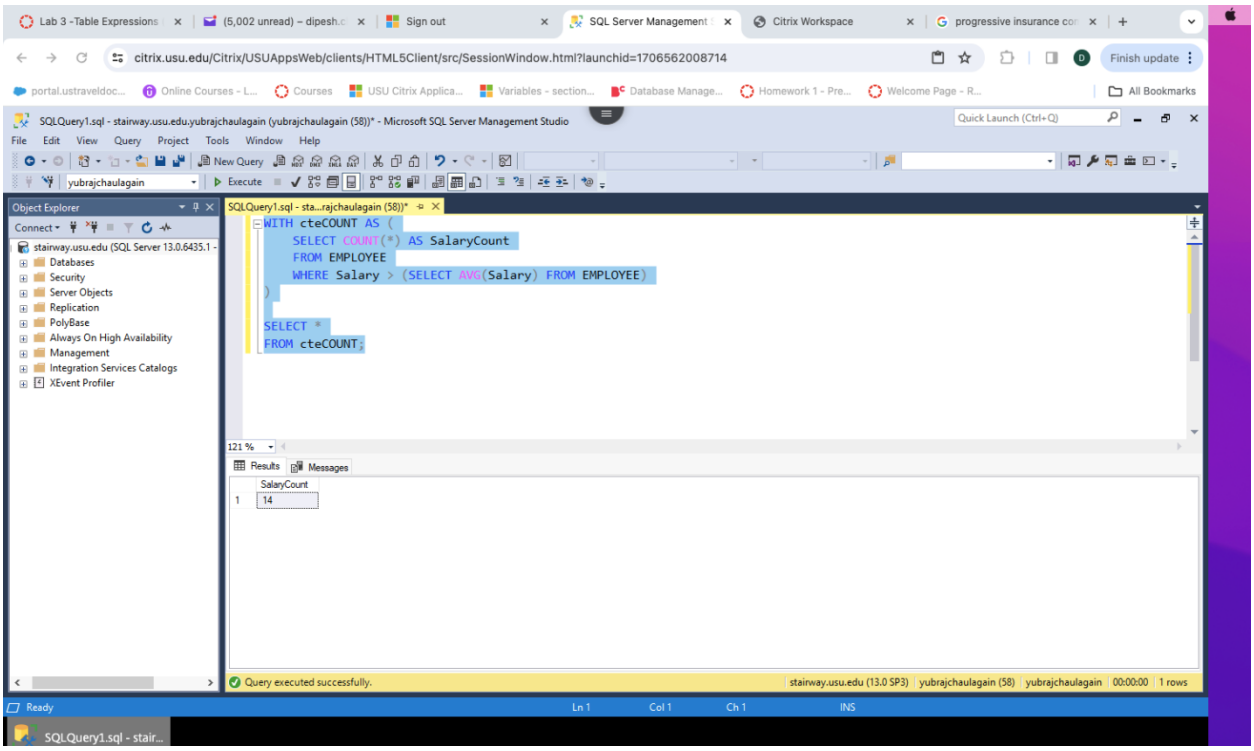
9. A Derived Table does not allow the 'ORDER BY' clause. Try the CTE below and see if it allows the ORDER BY clause? If the code won't run, rewrite (move the ORDER BY clause to a new location) so it works.

```
WITH cteState AS
    (SELECT State
        FROM CUSTOMER
        ORDER BY State)

SELECT *
FROM cteState;
```

New Code that will run

```
WITH cteState AS (
   SELECT State
   FROM CUSTOMER
)

SELECT *
FROM cteState
ORDER BY State
```

;




10. Does the following code use Inline or External Aliasing? Rewrite the code to demonstrate the 'other' (either Inline or External Aliasing)

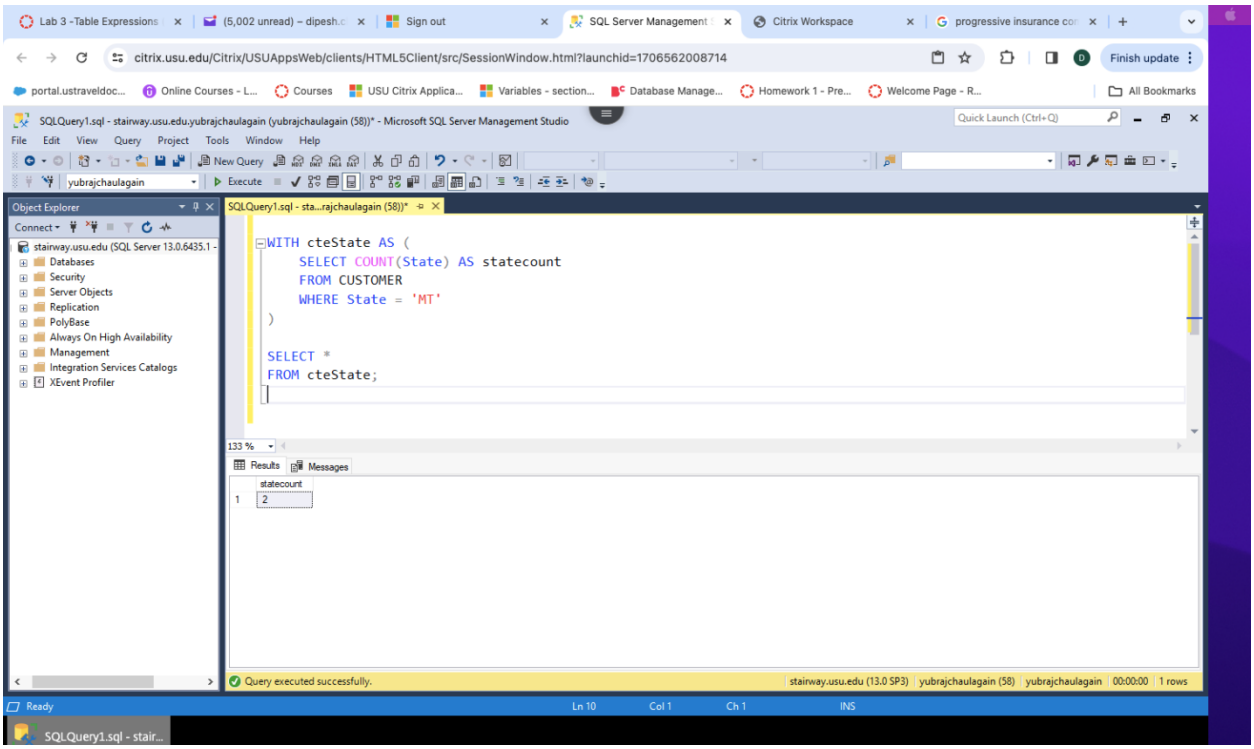
The provided code uses External Aliasing. External Aliasing involves giving a name to the CTE as a whole (not just the columns) using square brackets or double quotes.

```
WITH cteState ([statecount]) AS
    (SELECT COUNT(State)
        FROM CUSTOMER
        WHERE State = 'MT'
```

```
                )


SELECT *
FROM cteState;
```

Inline Aliasing:

```
WITH cteState AS (
    SELECT COUNT(State) AS statecount
    FROM CUSTOMER
    WHERE State = 'MT'
)


SELECT *
FROM cteState;
```




11. Henry has finished a class on SQL Server. He is now working for a company that uses Oracle. Describe how you would create a VIEW in Oracle.

CREATE VIEW view_name AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;

12. Create a VIEW called vCOUNT that uses a CTE called cteCOUNT to count the employees making over the average salary. Name your COUNT Alias SalaryCount. Add encryption to your VIEW.  *Hint: You need to include SELECT * FROM cteCOUNT in your VIEW.

Finally, type SELECT SalaryCount FROM vCOUNT to test your VIEW code after it's been created.

```
WITH cteCOUNT AS (
    SELECT COUNT(*) AS SalaryCount
    FROM EMPLOYEE
    WHERE Salary > (SELECT AVG(Salary) FROM EMPLOYEE)
)
CREATE VIEW vCOUNT
AS
SELECT * FROM cteCOUNT;
```

13. Conduct a brief Google search and identify 2-3 advantages of using: 1) derived tables, 2) common table expressions, and 3) views.  Provide a bulleted list of your findings.

**Derived Tables:**

Simplicity:

Derived tables are concise and can simplify complex queries by allowing you to create a temporary table within the scope of a larger query.

### Improved Readability:

They enhance query readability by encapsulating complex subqueries, making it easier to understand the main query's logic.

Performance Optimization:

In some cases, using derived tables can lead to optimized query execution plans, improving overall query performance.

## Common Table Expressions (CTEs):

Readability and Maintainability:

CTEs enhance code readability and maintainability by allowing you to break down a complex query into smaller, more manageable parts.

Recursive Queries:

CTEs support recursive queries, enabling the processing of hierarchical or tree-like data structures in a more elegant and efficient manner.

Reusable Code:

CTEs can be referenced multiple times within the same query, promoting code reuse and reducing redundancy.

## Views:

Abstraction of Complexity:

Views allow you to abstract complex underlying data structures, presenting a simplified and focused representation of the data to users or applications.

Security and Permissions:

Views provide a layer of security by allowing you to grant users access to specific views while restricting direct access to the underlying tables. This helps in controlling data access.

Consistency:

Views promote data consistency by offering a standardized way to access and present data. Changes to the underlying structure can be managed within the view, minimizing the impact on dependent queries.

These advantages contribute to better code organization, improved readability, and enhanced query performance in database development and maintenance.

***Subquery to be placed in Derived Table (Question 1 above)

```
SELECT COUNT(*) as SalaryCount
        FROM EMPLOYEE WHERE Salary > (SELECT AVG(Salary)
            FROM EMPLOYEE)
```

That’s it!  You did it!