

## DATA 4330 – Chapter 3 & 4 – Lab #2

Use the Antique Dataset to answer the following coding questions:

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

An example is provided below:

This lab incorporates the DPC Antiques dataset that is used throughout this module. Your instructor will provide you with the DDL (CREATE TABLE) and DML (INSERT INTO) text files necessary to load the database.

### Part 1 - Joins

1. Write a query that shows the highest employee commission broken down by gender. Don't include groups with less than \$100 max commission. (Hint: Commission is calculated by (quantity\*unitprice\*.1)).

SELECT

E.Gender,

MAX(SI.Quantity \* SI.UnitPrice \* 0.1) AS MaxCommission

FROM

SALE AS S

JOIN

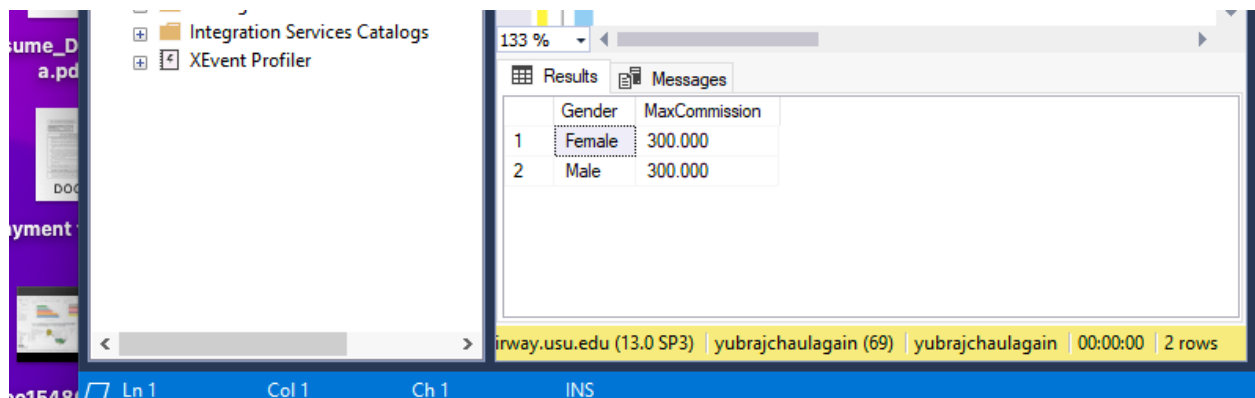
SALE\_ITEM AS SI ON S.SaleID = SI.SALEID JOIN EMPLOYEE AS E ON  
E.EmployeeID = S.EmployeeID

GROUP BY

E.Gender

HAVING

MAX(SI.Quantity \* SI.UnitPrice \* 0.1) >= 100;



	Gender	MaxCommission
1	Female	300.000
2	Male	300.000

2. List the last name of both the last name of the customer and employee where an employee sold an item to a customer with a last name starting with the letter 'H'. Use an appropriate alias even though you are not including an aggregate function.

```

SELECT

    C.LastName AS CustomerLastName,

    E.LastName AS EmployeeLastName

FROM

CUSTOMER as C

    JOIN

    SALE as S ON C.CustomerID = S.CustomerID

    JOIN

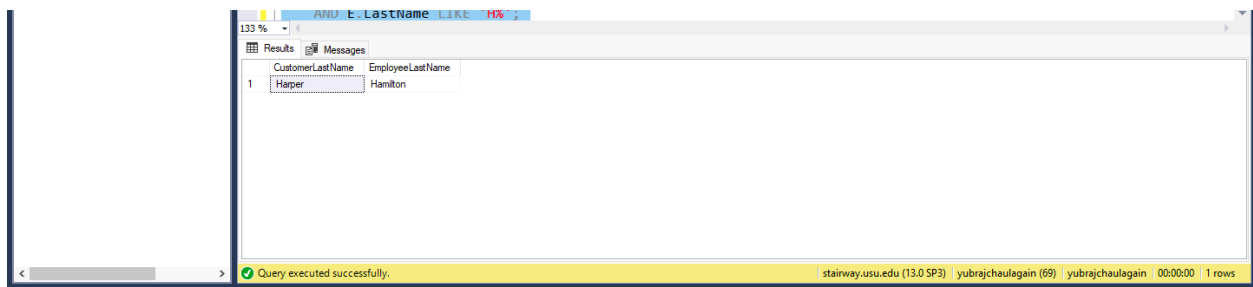
    EMPLOYEE as E ON S.EmployeeID = E.EmployeeID

WHERE

    C.LastName LIKE 'H%'

    AND E.LastName LIKE 'H%';

```



	CustomerLastName	EmployeeLastName
1	Harper	Hamilton

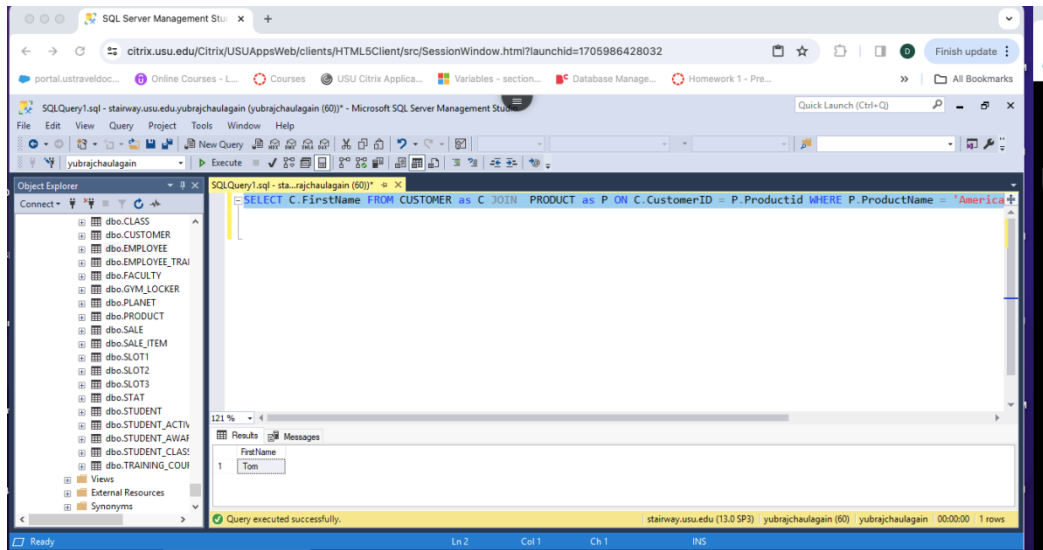
Query executed successfully.

- Write a query that shows the first name of the customer who bought a product with a product name of 'American Silver Eagle'. Do not include duplicate values.

```

SELECT C.FirstName FROM CUSTOMER as C JOIN PRODUCT as P ON
C.CustomerID = P.Productid WHERE P.ProductName = 'American Silver Eagle';

```



4. List the Last Name, First Name, and average commission of each employee. Sort from highest commission to lowest. Use an alias where appropriate. Make the commission output in currency format. If you encounter an error, it's probably related to the ORDER BY clause. Try doing an order by without using an alias. (Hint: This will require a 3 table join, Commission is calculated by (quantity\*unitprice\*.1).

SELECT

E.LastName,

E.FirstName,

FORMAT(AVG(SI.Quantity \* SI.UnitPrice \* 0.1), 'C', 'en-US') AS AverageCommission

FROM

Employee AS E

JOIN

Sale AS S ON E.EmployeeID = S.EmployeeID

JOIN

Sale\_Item AS SI ON S.SaleID = SI.SaleID

GROUP BY

E.LastName,

E.FirstName

ORDER BY

AVG(SI.Quantity \* SI.UnitPrice \* 0.1) DESC;

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with a tree view of the database schema, including tables like dbo.CLASS, dbo.CUSTOMER, and dbo.EMPLOYEE. The right pane shows a SQL query window with the following query:

```
SELECT  
    E.LastName,  
    E.FirstName,  
    FORMAT(AVG(SI.Quantity * SI.UnitPrice * 0.1), 'C', 'en-US') AS AverageCommission  
FROM  
    Employee AS E  
JOIN
```

Below the query, the 'Results' pane displays a table with 10 rows and 3 columns: LastName, FirstName, and AverageCommission. The data is as follows:

LastName	FirstName	AverageCommission
Fowler	Lucy	\$300.00
Bennett	Carl	\$215.00
Olsen	Beverly	\$150.00
Wendorf	Cub	\$113.33
Kelly	Samantha	\$110.00
Murray	Derek	\$105.00
Johnson	John	\$73.00
Matsko	Rich	\$70.00
Hill	Steve	\$60.00
Smith	Zach	\$25.00

The status bar at the bottom indicates 'Query executed successfully.' and '10 rows'.

5. -- **Write a count aggregate** with a Left OUTER JOIN to show each employees' last name, and the number of times they have enrolled in a

training course Use the EMPLOYEE and EMPLOYEE\_TRAINING Tables to code the solution.

SELECT

E.LastName,

FROM

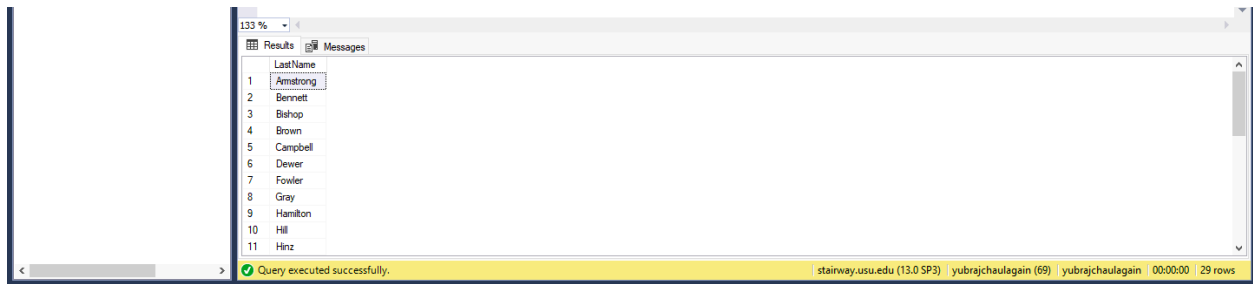
EMPLOYEE AS E

LEFT JOIN

EMPLOYEE\_TRAINING AS ET ON E.EmployeeID = ET.EmployeeID

GROUP BY

E.LastName;

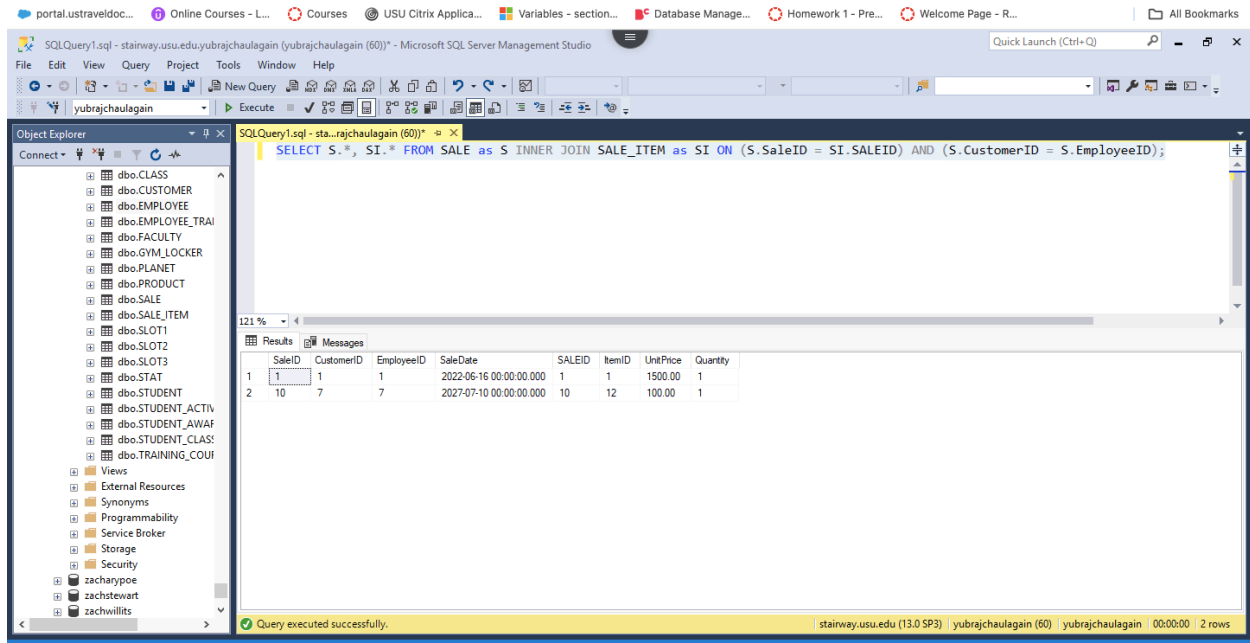


The screenshot shows a SQL query results window with a table containing 11 rows of last names. The window has a title bar with '133 %' and tabs for 'Results' and 'Messages'. The 'Results' tab is active, showing a table with one column 'LastName' and 11 rows of data. The status bar at the bottom indicates 'Query executed successfully.' and provides session information: 'stairway.usu.edu (13.0 SP3) yubrajchaulagain (69) yubrajchaulagain 00:00:00 29 rows'.

	LastName
1	Armstrong
2	Bennett
3	Bishop
4	Brown
5	Campbell
6	Dewer
7	Fowler
8	Gray
9	Hamilton
10	Hill
11	Hinz

6. First, write an INNER JOIN that connects the SALE and SALE\_Item tables. Then write an additional ON clause shows any sales where customer and employees IDs are the same to create a **COMPOSITE JOIN**.

```
SELECT S.*, SI.* FROM SALE as S INNER JOIN SALE_ITEM as SI ON (S.SaleID = SI.SALEID) AND (S.CustomerID = S.EmployeeID);
```



7. --Return customers who have not yet made a purchase (Use the Customer and Sale Table) and use a LEFT OUTER JOIN.

```
SELECT c.CustomerID, c.LastName, c.firstName
```

```
FROM CUSTOMER c LEFT OUTER JOIN SALE s ON c.CustomerID = s.CustomerID
```

```
WHERE s.CustomerID IS NULL;
```

	CustomerID	LastName	FirstName
1	2	Peterson	Dana
2	3	Holland	Scott
3	5	Miller	Maddy
4	6	Baum	Cody
5	8	Richardson	Ben
6	9	Brotherson	Duane
7	10	Stull	Katheline
8	11	Mills	Mille
9	14	Gonzler	Duane
10	15	Anderson	Maya
11	16	Douglas	Alex

Query executed successfully. | steinway.usu.edu (13.0 SP3) | yubrajchaulagain (144) | yubrajchaulagain | 00:00:00 | 33 rows

Sample Output:

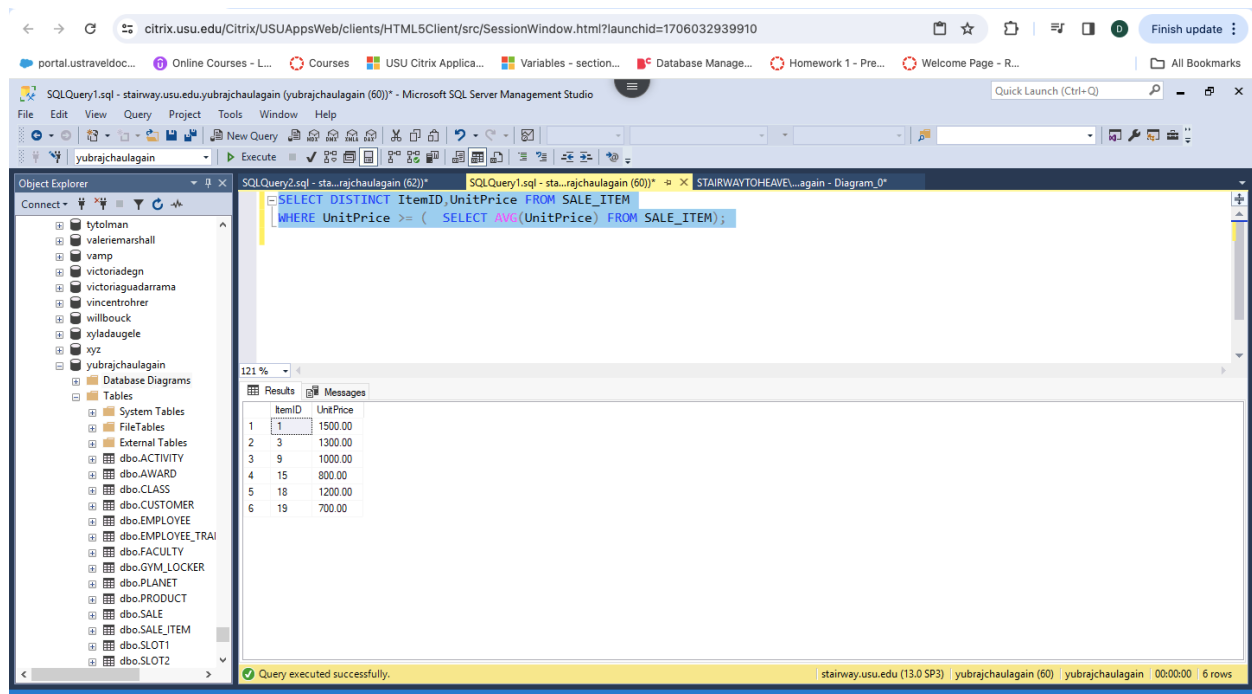
## Part 2 - Subqueries

- Write a Subquery that shows the product ID and unitprice price of any products that are GREATER than or EQUAL to the current average unit price. Don't include duplicate output.

```
SELECT DISTINCT ItemID,UnitPrice FROM SALE_ITEM
```



```
WHERE UnitPrice >= ( SELECT AVG(UnitPrice) FROM SALE_ITEM);
```



The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL statement:

```
SELECT DISTINCT ItemID, UnitPrice FROM SALE_ITEM  
WHERE UnitPrice >= ( SELECT AVG(UnitPrice) FROM SALE_ITEM);
```

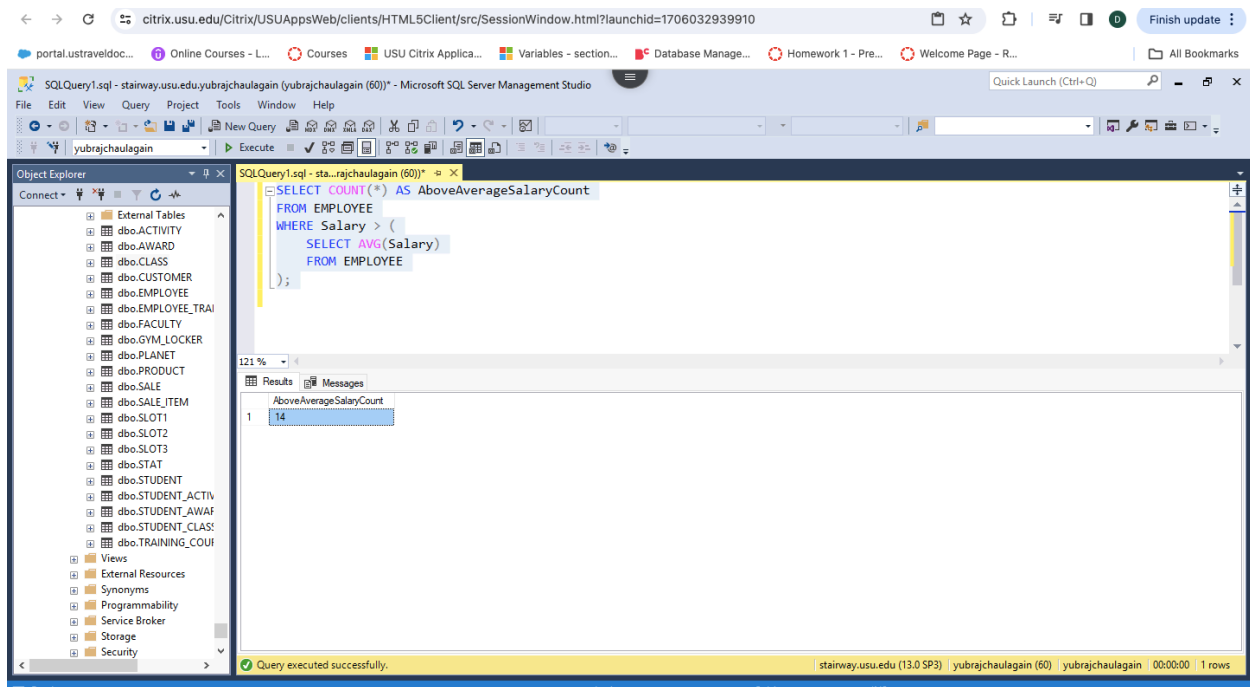
The query has been executed successfully, and the results are displayed in the Results pane. The results show 6 rows of data:

ItemID	UnitPrice
1	1500.00
2	1300.00
3	1000.00
4	800.00
5	1200.00
6	700.00

The status bar at the bottom indicates that the query was executed successfully and returned 6 rows.

9. Write an SQL statement that counts the number of employees that make an above average salary.

```
SELECT COUNT(*) AS AboveAverageSalaryCount FROM EMPLOYEE WHERE Salary  
> (  
  
SELECT AVG(Salary) FROM EMPLOYEE);
```



10. Write an SQL statement that displays gender, and separately counts and displays the number of male and female (use a GROUP BY) employees that make an above average salary. Don't include groups that are less than 2.

```
SELECT Gender, COUNT(*) AS EmployeeCount FROM EMPLOYEE WHERE Salary > (
    SELECT AVG(Salary) FROM EMPLOYEE
)
GROUP BY
```

Gender HAVING COUNT(\*) >= 2;

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window displays the following SQL query:

```
SELECT Gender, COUNT(*) AS EmployeeCount FROM EMPLOYEE WHERE Salary > (
    SELECT AVG(Salary) FROM EMPLOYEE
)
GROUP BY
    Gender HAVING COUNT(*) >= 2;
```

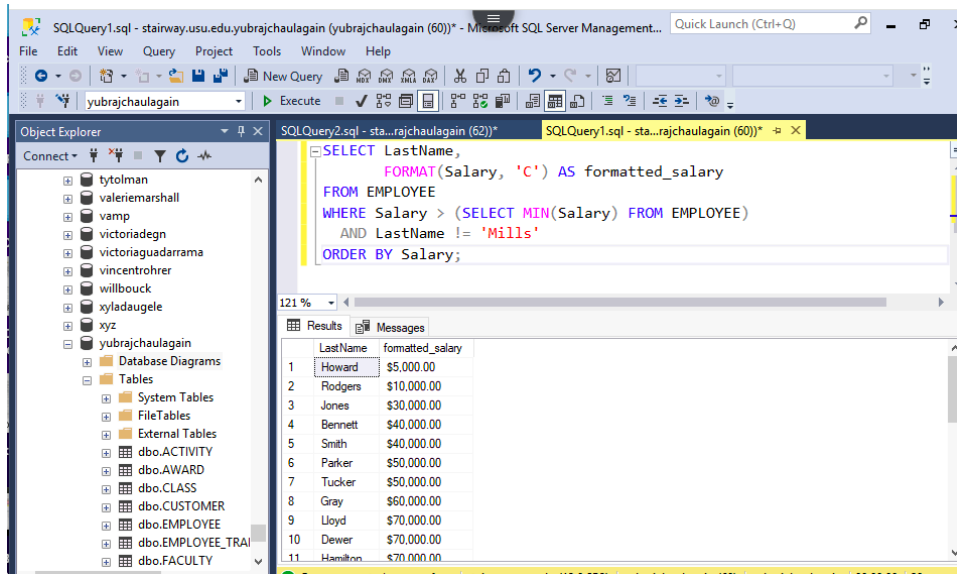
The Results pane shows the following data:

Gender	EmployeeCount
Female	7
Male	7

The status bar at the bottom indicates "Query executed successfully." and "2 rows".

11. Write a Subquery that shows the last name and salary of employees that make over \$25,000 the minimum salary of all employees. Don't include the employee with the last name 'Mills' in the output. Format the Salary output to currency (i.e., \$100,000.00) and order output from smallest to largest. If you encounter an error, it's probably related to the ORDER BY clause. Try doing an order by without using an alias.

```
SELECT LastName, FORMAT(Salary, 'C') AS formatted_salary
FROM EMPLOYEE
WHERE Salary > (SELECT MIN(Salary) FROM EMPLOYEE)
AND LastName != 'Mills'
ORDER BY Salary;
```



12. Write a Subquery that shows the first three letters of the employees' last name that earned a commission over 100. (Hint: (Quantity\*Unitprice\*.1)>=100).

```
SELECT LEFT(LastName, 3) AS first_three_letters FROM EMPLOYEE WHERE
EmployeeID IN (SELECT EmployeeID FROM SALE_ITEM WHERE (Quantity *
UnitPrice * 0.1) >= 100);
```

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The Object Explorer on the left displays a server named 'yubrajchaulagain' with various databases and tables. The SQL Query window in the center contains the following query:

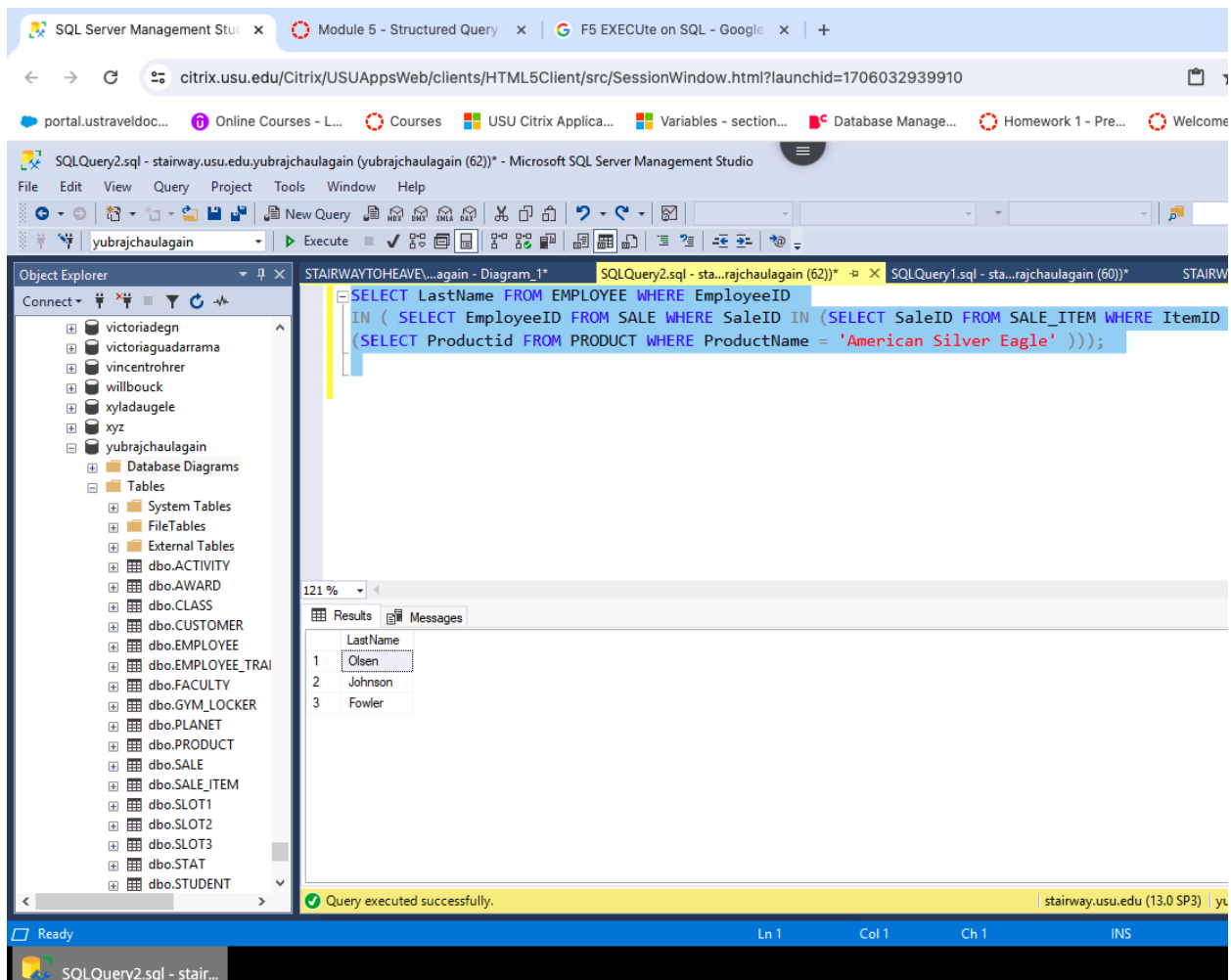
```
SELECT LEFT(LastName, 3) AS first_three_letters FROM EMPLOYEE
WHERE EmployeeID IN (SELECT EmployeeID FROM SALE_ITEM WHERE (Quantity > 0));
```

The Results pane at the bottom shows the output of the query, which is a list of employee last names truncated to three characters. The status bar at the bottom indicates that the query was executed successfully, returning 30 rows in 00:00:00.

first_three_letters
1 Ols
2 Mat
3 Tho
4 Mil
5 Joh
6 Smi
7 Wen
8 Bis
9 Hin
10 Dew
11 Hil

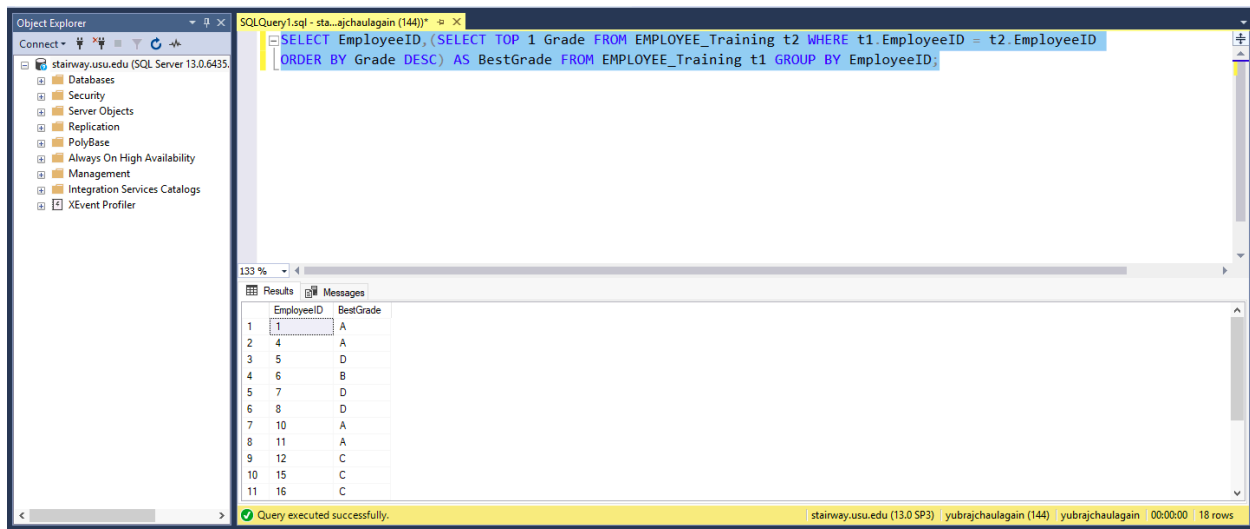
13. Write a Subquery that lists the Employees Last Name that have sold a product with the name “American Silver Eagle”. Question 7 above may provide some help with this question.

```
SELECT LastName FROM EMPLOYEE WHERE EmployeeID IN ( SELECT
EmployeeID FROM SALE WHERE SaleID IN (SELECT SaleID FROM SALE_ITEM
WHERE ItemID IN (SELECT Productid FROM PRODUCT WHERE ProductName =
'American Silver Eagle' )));
```



14. Write a **Correlated subquery** that shows the best grade earned by each student. Use the EMPLOYEE\_Training table to --code this problem. Use DISTINCT if needed. Output should include the employeeID and highest grade. Also, would you use MIN(GRADE) or MAX(GRADE)? \*Note, grades for this training only include A, B, C, D, and F.

```
SELECT EmployeeID,(SELECT TOP 1 Grade FROM EMPLOYEE_Training t2 WHERE
t1.EmployeeID = t2.EmployeeID ORDER BY Grade DESC) AS BestGrade FROM
EMPLOYEE_Training t1 GROUP BY EmployeeID;
```



Review the code and explain why it works. You might need to look up the ASCII function to better understand the solution.

The outer query is selecting the EmployeeID from the EMPLOYEE\_Training table and creating a subquery.

The subquery (SELECT TOP 1 Grade FROM EMPLOYEE\_Training t2 WHERE t1.EmployeeID = t2.EmployeeID ORDER BY ASCII(Grade) DESC) finds the top (highest) grade for each employee. It correlates with the outer query using t1.EmployeeID.

ORDER BY ASCII(Grade) DESC is used to order the grades based on their ASCII values in descending order. ASCII values are numeric representations of characters, so this ordering allows us to treat grades as numbers rather than characters.

ASCII(Grade) converts the character grade to its ASCII value. For example, the ASCII value of 'A' is lower than the ASCII value of 'B', so 'B' would be considered a higher grade.

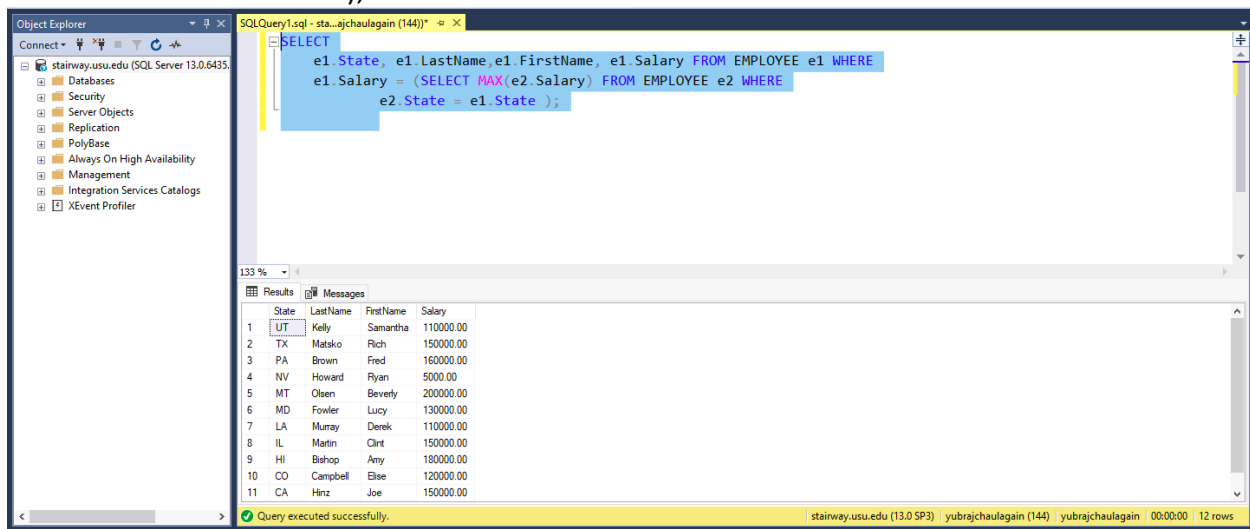
The outer query groups the results by EmployeeID.

This code works by using the ASCII function to convert character grades into their corresponding ASCII values and then ordering them in descending order. This allows the query to find the highest grade for each employee, considering grades as numeric values.

15. Write a **correlated subquery** that shows the highest salary by state. Use the EMPLOYEE table to code the problem. \* Your final WHERE clause in the correlated subquery DOESNT use

Employeeid at all in the problem... so don't write... (WHERE e1.EmployeeID = e2.EmployeeID).

```
SELECT e1.State, e1.LastName, e1.FirstName, e1.Salary FROM EMPLOYEE e1 WHERE  
e1.Salary = (SELECT MAX(e2.Salary) FROM EMPLOYEE e2 WHERE  
e2.State = e1.State );
```



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'stairway.usu.edu (SQL Server 13.0.6435)' server selected. The right pane shows a query window with the following SQL code:

```
SELECT e1.State, e1.LastName, e1.FirstName, e1.Salary FROM EMPLOYEE e1 WHERE  
e1.Salary = (SELECT MAX(e2.Salary) FROM EMPLOYEE e2 WHERE  
e2.State = e1.State );
```

Below the query window, the 'Results' pane displays the output of the query as a table with 11 rows and 4 columns: State, LastName, FirstName, and Salary.

State	LastName	FirstName	Salary
UT	Kelly	Samantha	110000.00
TX	Matsko	Rich	150000.00
PA	Brown	Fred	160000.00
NV	Howard	Ryan	5000.00
MT	Olsen	Beverly	200000.00
MD	Fowler	Lucy	130000.00
LA	Murray	Derek	110000.00
IL	Martin	Clint	150000.00
HI	Blahop	Amy	180000.00
CO	Campbell	Elise	120000.00
CA	Hinz	Joe	150000.00

The status bar at the bottom indicates 'Query executed successfully.' and '12 rows'.