

## **Competition**

Zachary C Schulte

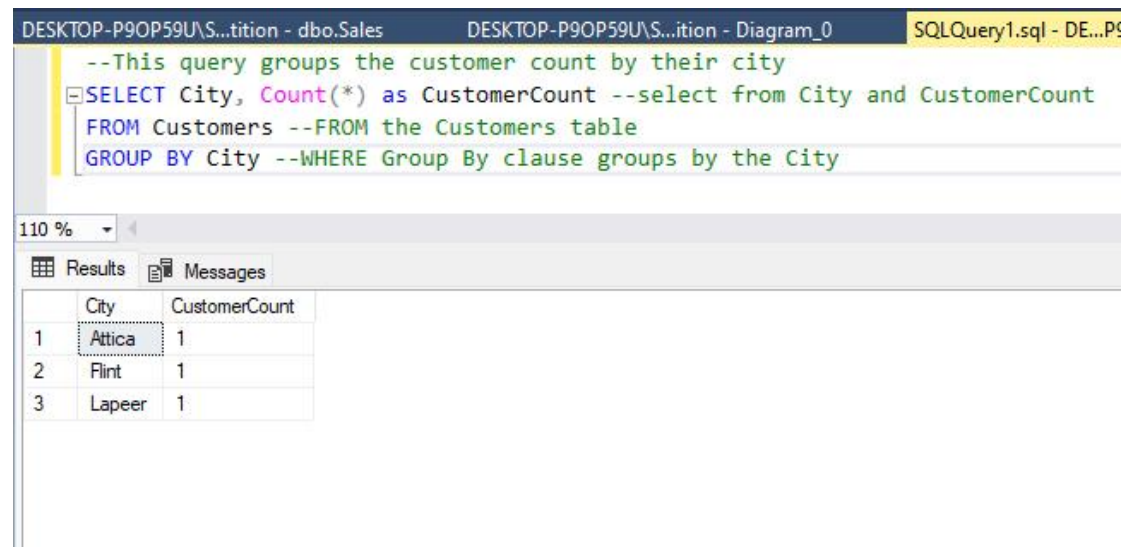
Information Security & Intelligence, Ferris State University

ISIN-325: Database Security

Professor Gogolin

December 7, 2023

1) Display how many purchases and the total amount purchased by each customer using one SQL statement.



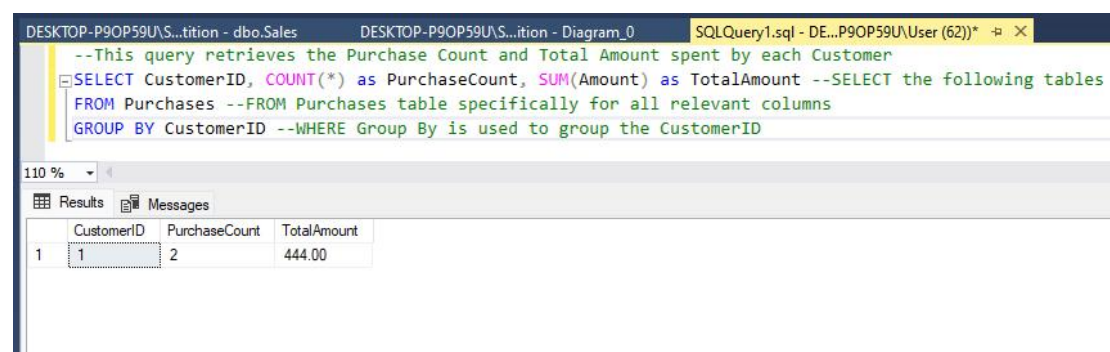
The screenshot shows a SQL query window with the following text:

```
--This query groups the customer count by their city  
SELECT City, COUNT(*) as CustomerCount --select from City and CustomerCount  
FROM Customers --FROM the Customers table  
GROUP BY City --WHERE Group By clause groups by the City
```

Below the query window, the 'Results' tab is active, displaying a table with the following data:

	City	CustomerCount
1	Attica	1
2	Flint	1
3	Lapeer	1

2) Display how many customers are in each city using one SQL statement.



The screenshot shows a SQL query window with the following text:

```
--This query retrieves the Purchase Count and Total Amount spent by each Customer  
SELECT CustomerID, COUNT(*) as PurchaseCount, SUM(Amount) as TotalAmount --SELECT the following tables  
FROM Purchases --FROM Purchases table specifically for all relevant columns  
GROUP BY CustomerID --WHERE Group By is used to group the CustomerID
```

Below the query window, the 'Results' tab is active, displaying a table with the following data:

	CustomerID	PurchaseCount	TotalAmount
1	1	2	444.00

3) Display all the sales each product is tied to, with the output sorted by product and category. [1]

The screenshot shows a SQL query window with the following text:

```
--SELECT uses the columns Category and ProductName using the MAX feature.
SELECT Category, MAX(ProductName) as ProductName
FROM Products -- FROM clause specifies the main table as Products
GROUP BY Category -- GROUP BY clause groups the data by Category
```

The Results pane shows the following data:

	Category	ProductName
1	Entertainment	TV
2	Fashion	T-Shirt
3	Gaming	PS5

4) Display all the products in each category. [1]

The screenshot shows a SQL query window with the following text:

```
--This query retrieves product information using the Order By clause on City, Category, and ProductName
SELECT City, Category, ProductName --SELECT the columns City, Category, and ProductName
FROM Products --FROM the source table Products
ORDER BY City, Category, ProductName --WHERE Order By clause is used to order the results by City, Category, and ProductName
```

The Results pane shows the following data:

	City	Category	ProductName
1	Attica	Entertainment	TV
2	Flint	Fashion	T-Shirt
3	Lapeer	Gaming	PS5

6) Display a count of the number of customers in each city sorted by city. [1]

The screenshot shows a SQL query window with the following text:

```
--This query selects the columns City and CustomerCount from the Customers table and groups and orders the data by City.
SELECT City, COUNT(*) as CustomerCount --SELECT City and CustomerCount as the columns
FROM Customers --FROM the source table Customers
GROUP BY City --Where the information is grouped by City
ORDER BY City --Where the information is ordered by City
```

The Results pane shows the following data:

	City	CustomerCount
1	Attica	1
2	Flint	1
3	Lapeer	1

7) Insert a new category "Health and Beauty Aids" with an appropriate description into the category table. (show me the sql and result set). [1]

The screenshot shows a SQL query window with the following text:

```
--Query to demonstrate inserting values into the Products table
INSERT INTO Products (ProductName, Category) VALUES ('Health and Beauty Aids', 'Health and Beauty Products')
--WHERE Insert Into is used to list the table and columns, along with the values being added into those columns
Select * From Products
```

Below the query window, the 'Results' tab displays the following data:

	ProductName	Category
1	PS5	Gaming
2	TV	Entertainment
3		
4	Health and Beauty Aids	Health and Beauty Products
5	Health and Beauty Aids	Health and Beauty Products

9) Display the category, purchase date and amount for each recipient include LName, FName) for each category sorted by LName that have purchased at least 4 products. [1]

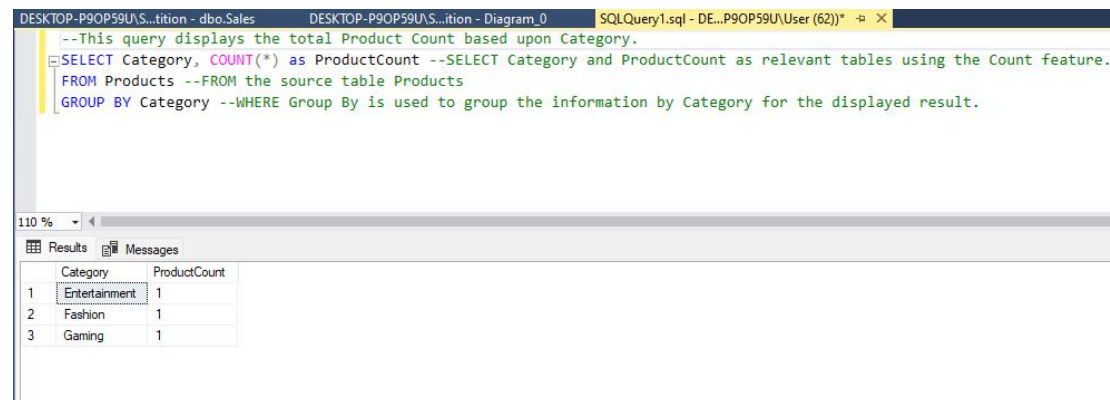
The screenshot shows a SQL query window with the following text:

```
--Query to demonstrate Customers that have bought more than 4 items
SELECT c.LastName, c.FirstName, p.Category, p.PurchaseDate, p.Amount
FROM Purchases p
JOIN Customers c ON p.CustomerID = c.CustomerID --Use Join to connect CustomerID from both tables
WHERE c.CustomerID IN (
    SELECT CustomerID
    FROM Purchases
    GROUP BY CustomerID
    HAVING COUNT(*) >= 4
) --Using COUNT to only allow data that presents a customer that has purchased 4 items or more
ORDER BY c.LastName
```

Below the query window, the 'Results' tab displays the following data:

	LastName	FirstName	Category	PurchaseDate	Amount
1	Schulte	Zach	Fashion	December 5th 2023	155.00
2	Schulte	Zach	Design	December 5th 2023	289.00
3	Schulte	Zach	Gaming	December 1st 2023	127.00
4	Schulte	Zach	Beauty	November 30th 2023	779.00

10) Display a count of the number of products for each category using one SQL statement. [1]



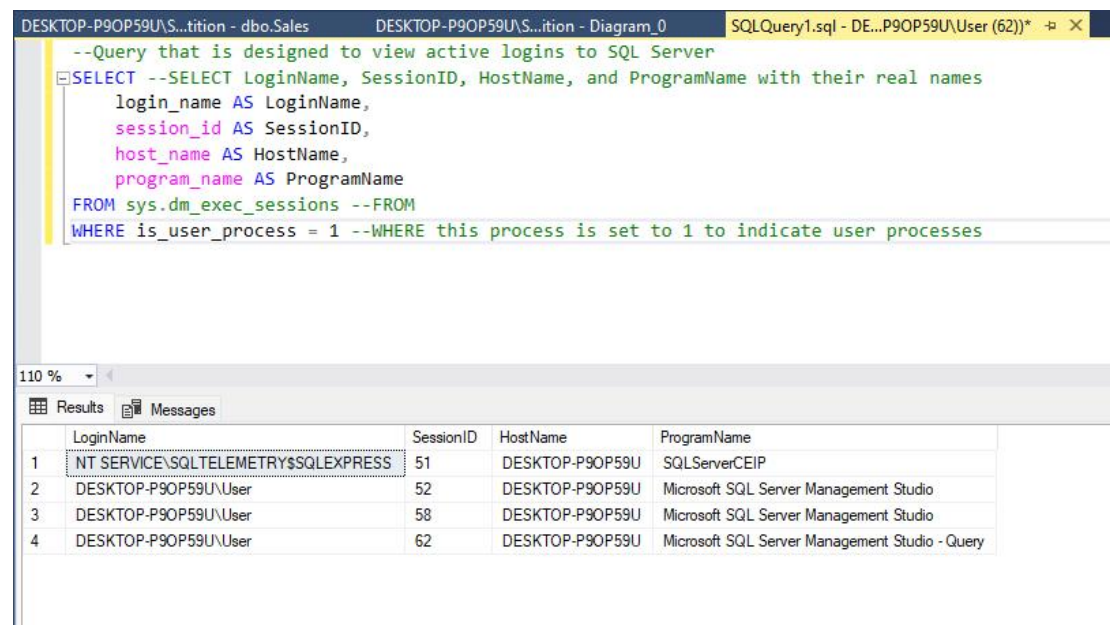
The screenshot shows a SQL query window with the following text:

```
--This query displays the total Product Count based upon Category.
SELECT Category, COUNT(*) as ProductCount --SELECT Category and ProductCount as relevant tables using the Count feature.
FROM Products --FROM the source table Products
GROUP BY Category --WHERE Group By is used to group the information by Category for the displayed result.
```

Below the query window, the Results pane displays the following table:

	Category	ProductCount
1	Entertainment	1
2	Fashion	1
3	Gaming	1

11) Display the active logins in SQL Server. [1]



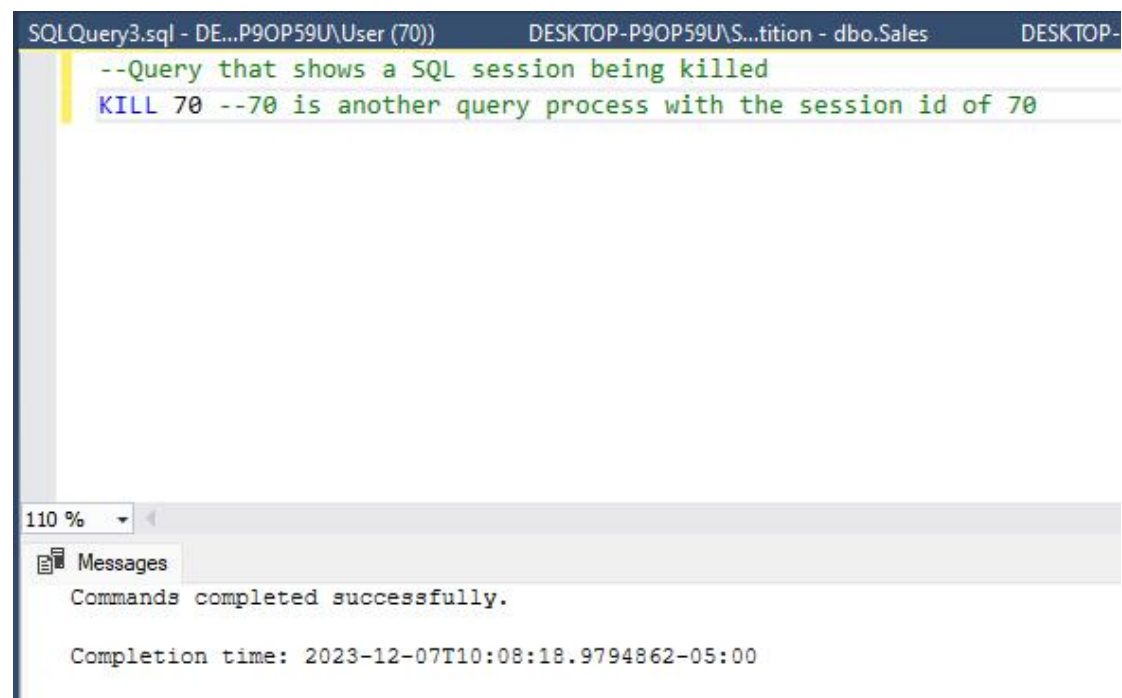
The screenshot shows a SQL query window with the following text:

```
--Query that is designed to view active logins to SQL Server
SELECT --SELECT LoginName, SessionID, HostName, and ProgramName with their real names
    login_name AS LoginName,
    session_id AS SessionID,
    host_name AS HostName,
    program_name AS ProgramName
FROM sys.dm_exec_sessions --FROM
WHERE is_user_process = 1 --WHERE this process is set to 1 to indicate user processes
```

Below the query window, the Results pane displays the following table:

	LoginName	SessionID	HostName	ProgramName
1	NT SERVICE\SQLTELEMETRY\$SQLEXPRESS	51	DESKTOP-P90P59U	SQLServerCEIP
2	DESKTOP-P90P59U\User	52	DESKTOP-P90P59U	Microsoft SQL Server Management Studio
3	DESKTOP-P90P59U\User	58	DESKTOP-P90P59U	Microsoft SQL Server Management Studio
4	DESKTOP-P90P59U\User	62	DESKTOP-P90P59U	Microsoft SQL Server Management Studio - Query

12) Demonstrate how to terminate a connection to SQL Server. [1]

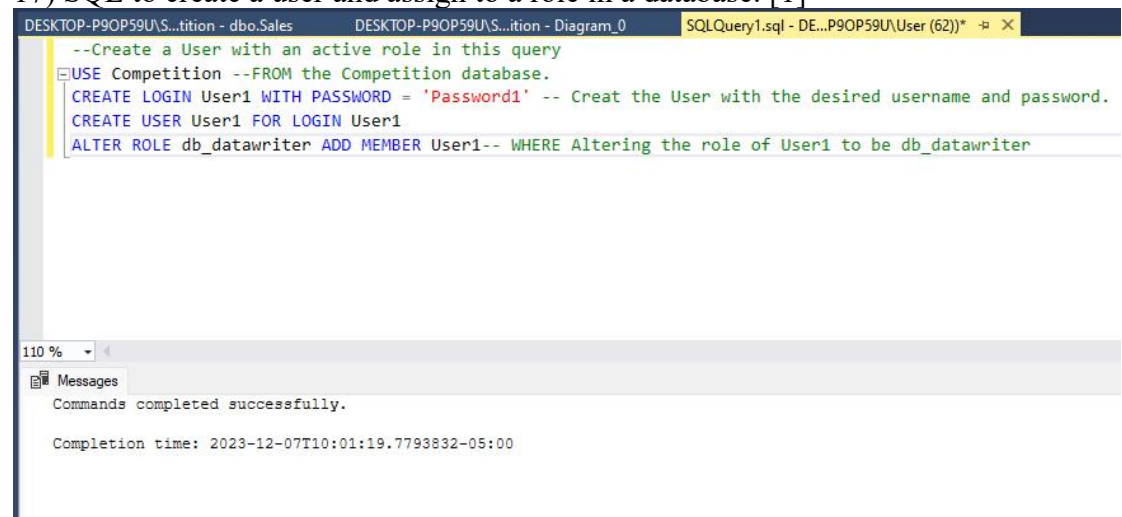


The screenshot shows a SQL Server Enterprise Manager window with the following details:

- Tab: `SQLQuery3.sql - DE...P9OP59U\User (70))`
- Server: `DESKTOP-P9OP59U\S...tition - dbo.Sales`
- Query Text:

```
--Query that shows a SQL session being killed
KILL 70 --70 is another query process with the session id of 70
```
- Zoom: 110 %
- Messages pane:
  - Commands completed successfully.
  - Completion time: 2023-12-07T10:08:18.9794862-05:00

17) SQL to create a user and assign to a role in a database. [1]



The screenshot shows a SQL Server Enterprise Manager window with the following details:

- Tab: `SQLQuery1.sql - DE...P9OP59U\User (62))`
- Server: `DESKTOP-P9OP59U\S...tition - Diagram_0`
- Query Text:

```
--Create a User with an active role in this query
USE Competition --FROM the Competition database.
CREATE LOGIN User1 WITH PASSWORD = 'Password1' -- Creat the User with the desired username and password.
CREATE USER User1 FOR LOGIN User1
ALTER ROLE db_datawriter ADD MEMBER User1-- WHERE Altering the role of User1 to be db_datawriter
```
- Zoom: 110 %
- Messages pane:
  - Commands completed successfully.
  - Completion time: 2023-12-07T10:01:19.7793832-05:00

18) Create an SQL routine that uses the Case statement. [1]

SQLQuery1.sql - DE...P9OP59U\User (58))\* X DESKTOP-P9OP59U\S...ition - Diagram\_0

```
-- SQL routine using the CASE statement
--UPDATE Products --FROM Products table
SET Status =
CASE --WHERE we are setting a When cluase to give a different status to each item based on Price
WHEN Price > 200 THEN 'High'
WHEN Price <= 100 AND Price > 60 THEN 'Medium'
ELSE 'Low'
END
Select * From Products
```

110 %

Results Messages

	ProductID	ProductName	Category	City	Price	Status
1	1	PS5	Gaming	Lapeer	300	High
2	2	TV	Entertainment	Attica	100	Medium
3	3	T-Shirt	Fashion	Flint	50	Low

19) Create an SQL routine that uses nested IF/ELSE/IF [1]

SQLQuery1.sql - DE...P9OP59U\User (58))\* X DESKTOP-P9OP59U\S...ition - Diagram\_0

```
--Query to create a variable and give it IF-ELIF-ELSE conditions
--DECLARE @Variable INT = 100 --Declare a variable that's an integer equal to 100
--If @Variable > 50 --First IF clause
PRINT 'Variable is greater than 50.'
--ELSE IF @Variable < 50 --Elif clause
PRINT 'Variable is less than to 50.'
ELSE --If neither IF or ELIF work
PRINT 'Variable is 50.'
```

110 %

Messages

Variable is greater than 50.

Completion time: 2023-12-07T10:30:38.4528237-05:00



## 20) Create an SQL routine that uses at temp table. [2]

DESKTOP-P9OP59U\...on - dbo.Products    SQLQuery1.sql - DE...P9OP59U\User (58))\*    X

```
-- SQL query using a temp table

CREATE TABLE #TempTable (
    ID INT,
    Name NVARCHAR(50)
) --WHERE Create Table is used with a # to indicate a temporary table

INSERT INTO #TempTable (ID, Name) --WHERE Insert Into is used in the temp table to provide information
VALUES
    (1, 'John'),
    (2, 'Jane'),
    (3, 'Michael')

Select * FROM #TempTable
```

110 %

Results    Messages

	ID	Name
1	1	John
2	2	Jane
3	3	Michael

## 21) SQL that joins tables that are located in two databases. [1]

DESKTOP-P9OP59U\...S.Shoe - dbo.Shoe    DESKTOP-P9OP59U\...Shoe - Diagram\_0    SQLQuery1.sql - DE...P9OP59U\User (58))\*    X    DESKTOP-P9OP59U\...ition - Diagram\_0

```
-- SQL inner joining tables from two databases
--Combines both Shoe table rows and Product table rows
SELECT *FROM Shoe.dbo.Shoe AS T1 --SELECT From Shoe database select the Shoe table
INNER JOIN Competition.dbo.Products AS T2 ON T1.ProductID = T2.ProductID --WHERE Inner join between Competition and Shoe happens
```

110 %

Results    Messages

ShoeID	EUSize	USSize	Gender	WPrice	RPrice	ReleaseDT	BrandID	Name	ProductID	ProductID	ProductName	Category	City	Price	Status
--------	--------	--------	--------	--------	--------	-----------	---------	------	-----------	-----------	-------------	----------	------	-------	--------



28) Create a routine that demonstrates characteristics of Union [1]

DESKTOP-P9OP59U\S.Shoe - dbo.Shoe    DESKTOP-P9OP59U\S.Shoe - Diagram\_0    SQLQuery1.sql - DE...P9OP59U\User (58)\*    DESKTOP-P9OP59U\S...ition - Di

```
--Query to demonstrate the characteristics of UNION
SELECT Category, City FROM Products --WHERE Category and City are being taken from the Products table
UNION
SELECT Category, ProductName FROM Sales--WHERE Category and ProductName are being taken from the Sales table
```

110 %

Results    Messages

	Category	City
1	Entertainment	Attica
2	Entertainment	iPad
3	Entertainment	TV
4	Fashion	Flint
5	Fashion	T-Shirt
6	Food	Chicken Breast
7	Gaming	Lapeer
8	Gaming	PS5

29) Create a routine that uses Select Into to populate a table from another table. [1]

DESKTOP-P9OP59U\S.Shoe - dbo.Shoe    DESKTOP-P9OP59U\S.Shoe - Diagram\_0    SQLQuery1.sql - DE...P9OP

```
--Query to demonstrate a 'Select Into' statement
SELECT TOP 100 --Grab the top 100 from two columns using a select statement
LangID, LangName
INTO NewLanguageTable --WHERE Into places the information into a new table
FROM Language --FROM the Language table
```

110 %

Messages

(34 rows affected)

Completion time: 2023-12-07T10:45:54.2196475-05:00

30) Create a SQL routine that uses a subquery. [1]

DESKTOP-P9OP59U\S.Shoe - dbo.Shoe    DESKTOP-P9OP59U\S.Shoe - Diagram\_0    SQLQuery1.sql - DE...P9OP59U\User (58)\*    DESKTOP-P9OP59U\S

```
--Query to demonstrate a subquery of MAX SalesCount from Sales using the Products table column Category
SELECT Category, (SELECT MAX(SalesCount) FROM Sales) AS MaxValue
FROM Products
```

110 %

Results    Messages

	Category	MaxValue
1	Gaming	20
2	Entertainment	20
3	Fashion	20