**IS-4420 Database Fundamentals**

**Lab 5**

* SQL SELECT Queries
* Advanced SQL SELECT Queries

20 points

**Summary**

In Lab 5, we will practice writing SELECT queries. Please be sure to review the PowerPoints, videos, and Zoom webinar for reference.

The deliverable for the lab is your SELECT queries to answer the below business questions.

**Setup**

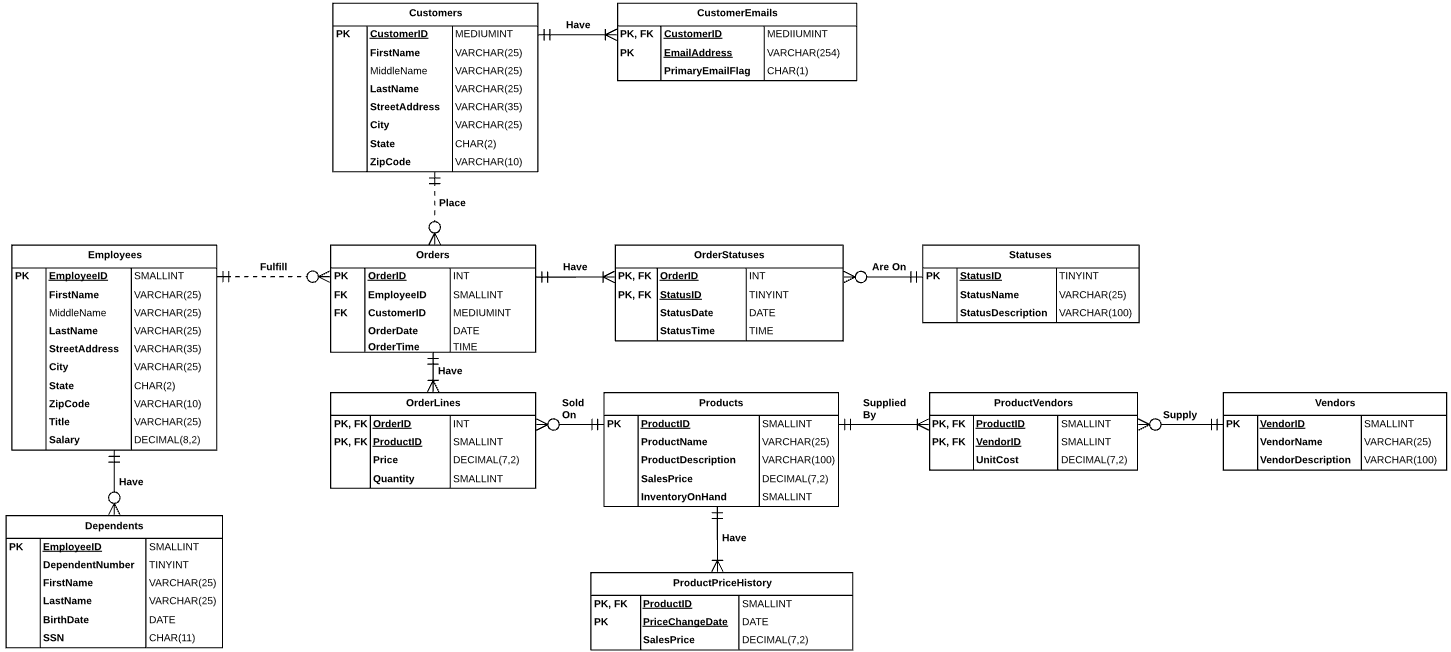
1. Connect to the SQL lab environment.
   1. If you have not yet connected to the SQL lab environment, follow the steps outlined in the **Connecting\_To\_IS4420\_RDS\_WebAccess.docx** file under the SQL Lab Setup module.
   2. Launch SQL Server Management Studio by clicking the Windows charm and typing SSMS. Change the authentication method from Windows to SQL, and then type Swoop for the username and Swoop for the password.
2. Retrieve the **Lab\_5\_Setup.sql** file that was provided.
3. Change the first 3 lines from Lab\_5\_<StudentName> to be your name.
   1. Example Lab\_5\_JacobCase.
4. Run the script to create the database for the lab.
5. Refresh the Databases menu on the left. Your Lab\_5\_<StudentName> database should be created.

**Note**

* You are encouraged to collaborate on homework assignments with your peers.
* Please refer to the PowerPoints & videos for SQL syntax.
* If you get stuck, attend tutoring office hours or send questions via email to the TAs.
* Assignments will take time, start them early.
* Class attendance will be rewarded by getting hints in class about the assignments.

**1: SELECT Queries (16 points)**

Use the below ERD for reference while writing your SQL queries:



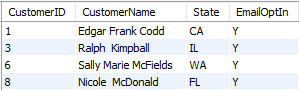
1. Show CustomerID, CustomerName (concatenate FirstName, MiddleName, & LastName with spaces between), State, and EmailOptIn for all Customers who do not live in New Mexico (NM) and are opted in to receive marketing emails. Order the results ascending by CustomerID

(1 point)

Hints:

* 1. This query requires no joins.
  2. Remember, NULLs cause the whole expression to result in NULL if not handled correctly.
  3. Refer to the SQL PowerPoints for how to handle NULLs.

#1 Result Set



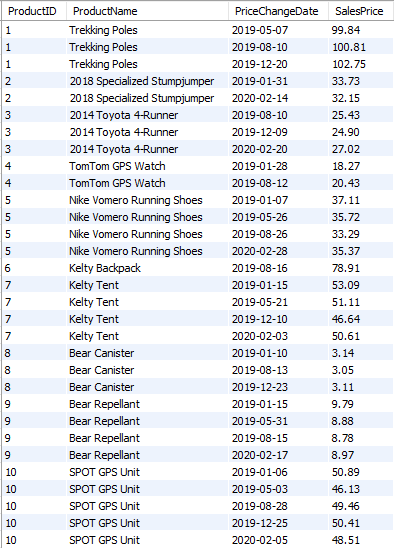
1. Show ProductID, ProductName, PriceChangeDate, and the SalesPrice that corresponds with the price change date.

(1 point)

Hints:

* 1. First, write a simple query that SELECTs ProductID & ProductName from Products.
  2. Add a join to the ProductPriceHistory table.
  3. Add the PriceChangeDate and the SalesPrice columns from the ProductPriceHistory table to the SELECT list.

#2 Result Set



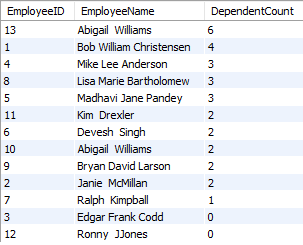
1. Show the EmployeeID, EmployeeName (concatenate FirstName, a space, and LastName), and a column called DependentCount showing how many Dependents each Employee has. Sort the results descending by how many Dependents each Employee has. If an Employee has no dependents, the DependentCount column should show 0.

(1 point)

Hints:

* 1. First, write a simple query that SELECTs EmployeeID and the concatenated EmployeeName columns from the Employees table.
  2. Next, add a join into the Dependents table matching on EmployeeID.
     1. Which type of join do you need to do to make sure the results include Employees with no Dependents?
  3. Add the COUNT() aggregation and alias it as DependentCount.
  4. Add the necessary GROUP BY clause.
  5. Finally, add the ORDER BY clause to sort the results descending by COUNT() expression.

#3 Result Set



1. Show CustomerID, FirstName, and LastName for any Customers who don’t have an email address (you must conduct a join to earn points for this question).

(1 point)

Hints:

* 1. Start with a simple query that SELECTs CustomerID, FirstName, and LastName FROM the Customers table.
  2. Add a join into the CustomerEmails table.
     1. Which type of join should you use here if you want to include rows from the Customers table where there is no match in the CustomerEmails table?
  3. Add a WHERE clause that will filter out any rows where there is not a matching row in the CustomerEmails table.

#3 Result Set



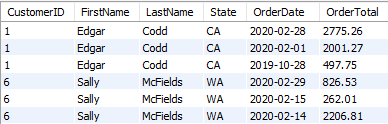
1. Show CustomerID, FirstName, LastName, State, OrderDate, and OrderTotal for any Customers in Washington (‘WA’) or California (‘CA’). Use the IN clause to filter the rows on State. Order the results ascending by CustomerID and descending by OrderDate.

(2 points)

Hints:

* 1. Start with a simple query that SELECTs CustomerID, FirstName, LastName, and State from Customers.
  2. Add a WHERE clause with the IN keyword to filter rows on State to only include ‘WA’ or ‘CA’.
  3. Join to Orders matching on CustomerID.
  4. Join to OrderLines matching on OrderID.
  5. Add OrderDate to the SELECT list.
  6. Add a SUM() column that sums Price \* Quantity and alias it as OrderTotal.
  7. Add a GROUP BY clause that includes all columns aside from the SUM() column.
  8. Finally, add the ORDER BY clause.

#4 Result Set



1. Show the EmployeeID, EmployeeName (concatenate FirstName, MiddleName, LastName), State as EmployeeState, DependentName (concatenate FirstName & LastName), BirthDate, and Age (calculated) for all Employees who live in Colorado (‘CO’). Sort the results descending by BirthDate.

(2 points)

Hints:

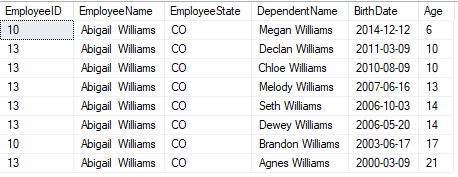
* 1. Start with a simple query that SELECTs EmployeeID, EmployeeName (concatenated), & State form the Employees table. Add the WHERE clause to include only Employees from Colorado (‘CO’).
  2. Add a join into the Dependents table matching on EmployeeID.
  3. Add DependentName (concatenated) and the BirthDate column.
  4. Add a column called Age column by using the DATEDIFF() function.

Use this Microsoft page for reference about the DATEDIFF() function:

<https://docs.microsoft.com/en-us/sql/t-sql/functions/datediff-transact-sql?view=sql-server-ver15>

* 1. Finally, add the ORDER BY clause to sort the results by BirthDate descending.

#5 Result Set



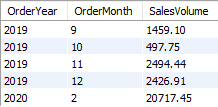
1. Show OrderYear, OrderMonth, and SalesVolume for all orders. Sort the results ascending by OrderYear and OrderMonth.

(2 points)

Hints:

* 1. Start with a simple query that SELECTs OrderDate FROM Orders.
  2. Replace the OrderDate column with 2 columns, 1 aliased as OrderYear and the other aliased as OrderMonth that respectively use the YEAR() and MONTH() functions to derive the order year and month from the OrderDate column.
  3. Add a join into the OrderLines table.
  4. Add a column that sums Price \* Quantity to calculate SalesVolume.
  5. GROUP BY OrderYear and OrderMonth.
  6. Finally, ORDER BY OrderYear ascending and OrderMonth ascending.

#7 Result Set



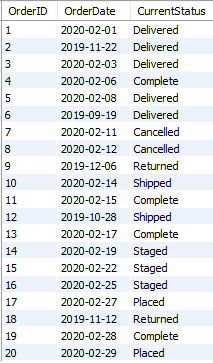
1. Show the OrderID, OrderDate and the current status of each Order. Use a correlated subquery in the SELECT list to get the Status with the most recent Status Date & Time and alias it as CurrentStatus.

(2 points)

Hints:

* 1. First, write a simple query that shows OrderID and OrderDate for each Order. Sort the results ascending by OrderID.
  2. Put a comma after OrderDate, hit enter, and type “() AS CurrentStatus” on the next line. The correlated subquery will go inside the parentheses.
  3. Put the cursor inside the parentheses and hit enter a few times. Write a query that SELECTs StatusName FROM Statuses and joins to OrderStatuses on StatusID. The results should be sorted descending by StatusDate and StatusTime. Add “TOP 1” at the beginning of the SELECT list in the subquery, which will ensure the subquery will return 1 row for each Order- the most recent status. Finally, add a WHERE clause to the subquery to match (“correlate”) the outer query with the inner query on OrderID.

#8 Result Set



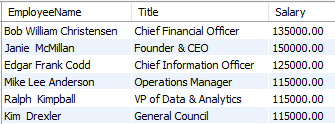
1. Show EmployeeName (concatenated), Title, and Salary of all Employees whose salary is above average for the company. You must use a subquery for full credit.

(2 points)

Hints:

* 1. Start with a simple query that SELECTs EmployeeName (concatenated), Title, and Salary FROM the Employees table.
  2. Add a WHERE clause that compares Salary with the result of a subquery.
  3. Inside the parentheses, write a subquery that returns the average for salary for all rows in the Employees table.

#9 Result Set



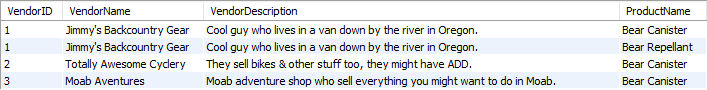
1. Show VendorID, VendorName, VendorDescription, and the ProductName for any Vendors who sell Products that help you stay safe from bears in the backcountry. To identify such Products, search for any Product with a name that starts with “Bear.”

(2 points)

Hints:

* 1. Write a simple query that SELECTs VendorID, VendorName, VendorDescription from the Vendors table.
  2. Add a JOIN to the ProductVendors table.
  3. Add a JOIN to the Products table.
  4. Add a WHERE clause that uses the LIKE operator to include rows that start with “Bear.”
  5. Add an ORDER BY clause to sort the result set by VendorID ascending.

#10 Result Set



**3: Review (4 points)**

**Answer the following questions:**

1. What is the difference between Calculated Columns vs Aggregations?

**[answer here]**

1. What is the GROUP BY clause used for? Provide an example of when you would use it.

**[answer here]**

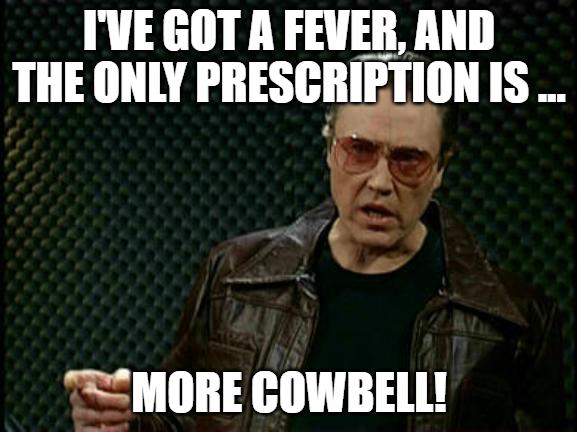
1. Describe what Boolean (Predicate) Logic is, and how it is used to filter rows from the result set.

**[answer here]**

1. Describe the difference between Joins vs Set Operators. Describe the criteria for datasets to be combined when using set operators.

**[answer here]**

**4: Extra Credit (More Cowbell!) (4 points)**

****

[**https://www.youtube.com/watch?v=cVsQLlk-T0s**](https://www.youtube.com/watch?v=cVsQLlk-T0s)

Write a query that will be used for invoices. The query needs to show Order Line level information, and then the final row will show the Order Totals. To earn the extra credit, you must use the UNION set operator.

The OrderLine level columns should include OrderID, OrderDate, ProductName, Price, Quantity, and LineTotal.

The Order Total level information should include OrderID, OrderDate, and OrderTotal. For ProductName, show “Order Total.” Align OrderTotal with LineTotal.

Hints:

1. Write a query that shows 1 row for each Order Line from OrderID 6 with the columns specified above.
2. Write a separate query with 1 row showing the Order Total with the columns specified above.
3. Make sure the queries are Union Compatible (see the Joins PowerPoint).
4. Combine the queries by adding the UNION keyword between them.

Extra Credit Result Set

**Lab 5 Deliverable:**

Place all your SELECT queries in a .sql file called Lab\_5\_StudentName.sql. Put a single line comment on the line prior to each query indicating which number the query corresponds with.

Please list them in order corresponding to the assignment.

For example:

-- Question 1

SELECT …

FROM …

-- Question 2

SELECT …

FROM …

Add your answers to the Review questions in multi-line comments.

For example:

/\*

1- What is the difference between Calculated Columns vs Aggregations?

The difference between Calculated Columns vs Aggregations is …

2- What is the GROUP BY clause used for? Provide an example of when you would use it.

\*/