

**CSE 581 : INTRODUCTION TO DATABASE MANAGEMENT SYSTEMS****LAB : 5****DATE : 10/5/2022**

**Q1)** Write a SELECT statement that returns two columns based on the Customers table. The first column, ContactName, is the customer's name in this format: Customers first name (i.e. CustomerFirst column) followed by first letter of Customers last name followed by a dot. The second column, ContactPhone, is the CustPhone column without the area code. Only return rows for those customers in the area codes that start with a digit of 5. Sort the results set by Customers first name in ascending order. *Use Examples database.*

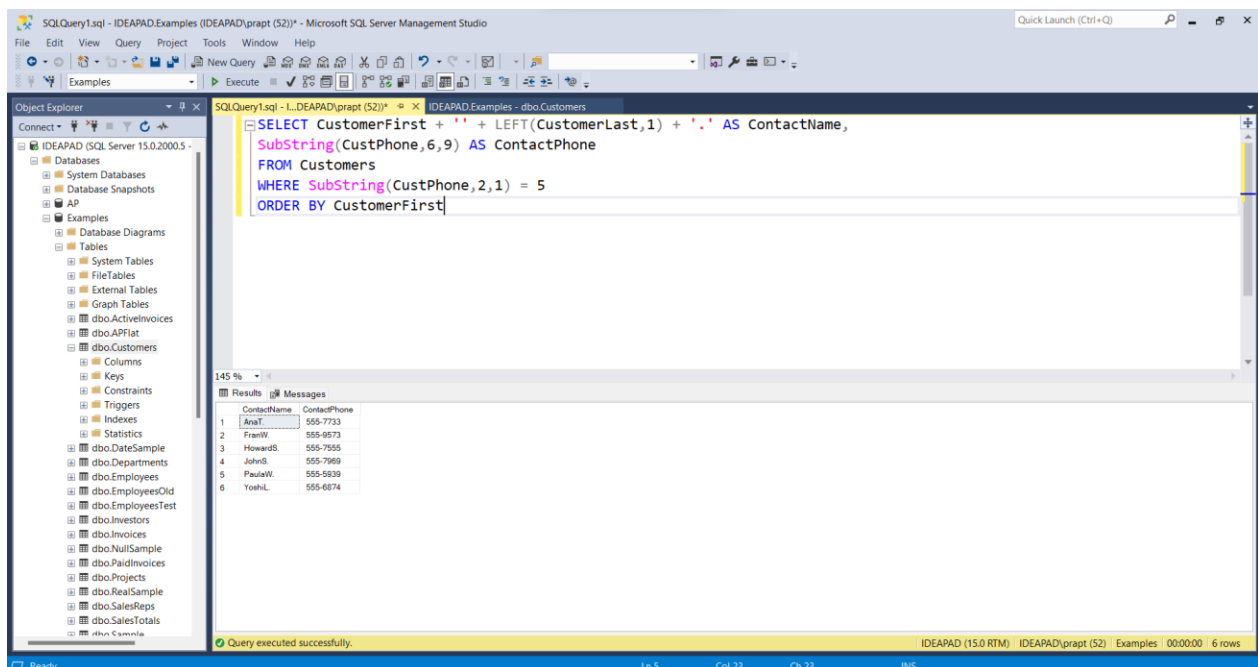
**Ans:** SELECT CustomerFirst + ' ' + LEFT(CustomerLast,1) + '.' AS ContactName,

SubString(CustPhone,6,9) AS ContactPhone

FROM Customers

WHERE SubString(CustPhone,2,1) = 5

ORDER BY CustomerFirst



**Comment:** Here, SELECT statement is used to display two columns from the Customers table. For the first column, CustomerFirst column is concatenated with the first letter

CustomerLast column followed by a '.' at the end by using '+' operator. The first letter of the CustomerLast column is obtained by using the LEFT() function that takes the column name i.e., CustomerLast and number of characters to be selected from the left i.e., 1 as parameters. For the second column, to remove the area code of the CustPhone column, SubString() function is used with parameters as the column name i.e., CustPhone, starting index as 6 and the length as 9. Records to be displayed are such that the area codes of those customers start with a '5'. WHERE clause is used to get these records with the use of SubString() function with parameters: CustPhone, 2 as start and 1 as length. The result set is then ordered by CustomerFirst in ascending order using ORDER BY.

**Remark:** SubString(ColumnName, startindex, length) function is used to get certain length of characters from a string.

Left(ColumnName, length) function is used to get certain length of characters from the left of the string.

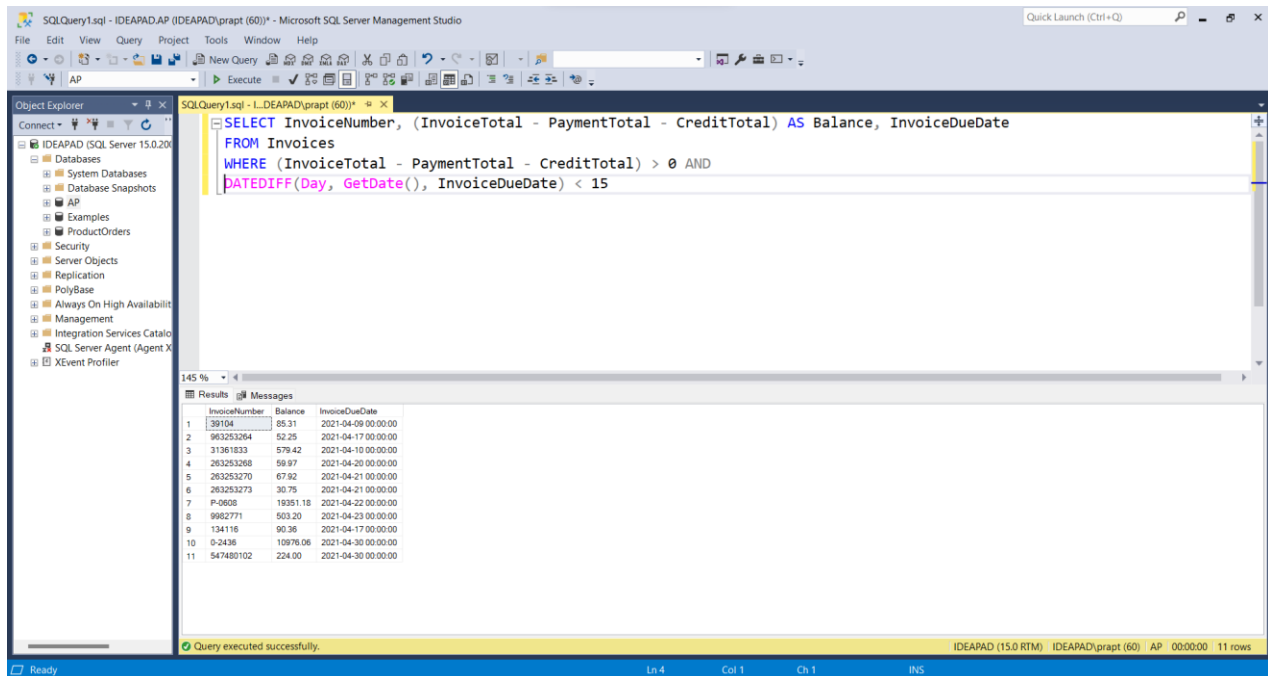
**Q2)** Write a SELECT statement that returns the InvoiceNumber and balance due for every invoice with a non-zero balance and an InvoiceDueDate that's less than 15 days from today. Use AP database.

**Ans:** SELECT InvoiceNumber, (InvoiceTotal - PaymentTotal - CreditTotal) AS Balance, InvoiceDueDate

FROM Invoices

WHERE (InvoiceTotal - PaymentTotal - CreditTotal) > 0 AND

DATEDIFF(Day, GetDate(), InvoiceDueDate) < 15



**Comment:** Here, SELECT statement is used to display InvoiceNumber column, Balance column which is the result of (InvoiceTotal – PaymentTotal – CreditTotal) and InvoiceDueDate column from the Invoices table. Records are selected by putting a condition using the WHERE clause such that the Balance is not zero i.e., greater than zero which is done using the '>' operator and the InvoiceDueDate that is less than 15 days from today which is done using the DATEDIFF(interval, date1, date2) function. DATEDIFF() takes parameters like interval which we choose as 'Day' as we want the interval of 15 days, date1 which we choose as GetDate() which takes current system's date and time and date2 which we choose as InvoiceDueDate to take the difference in days between today's date and the InvoiceDueDate which should be less than 15.

**Remark:** DATEDIFF(interval, date1, date2) is a function to get the difference between two dates in different interval formats like month, year, day etc.

GetDate() function gets the current database system's date and time.

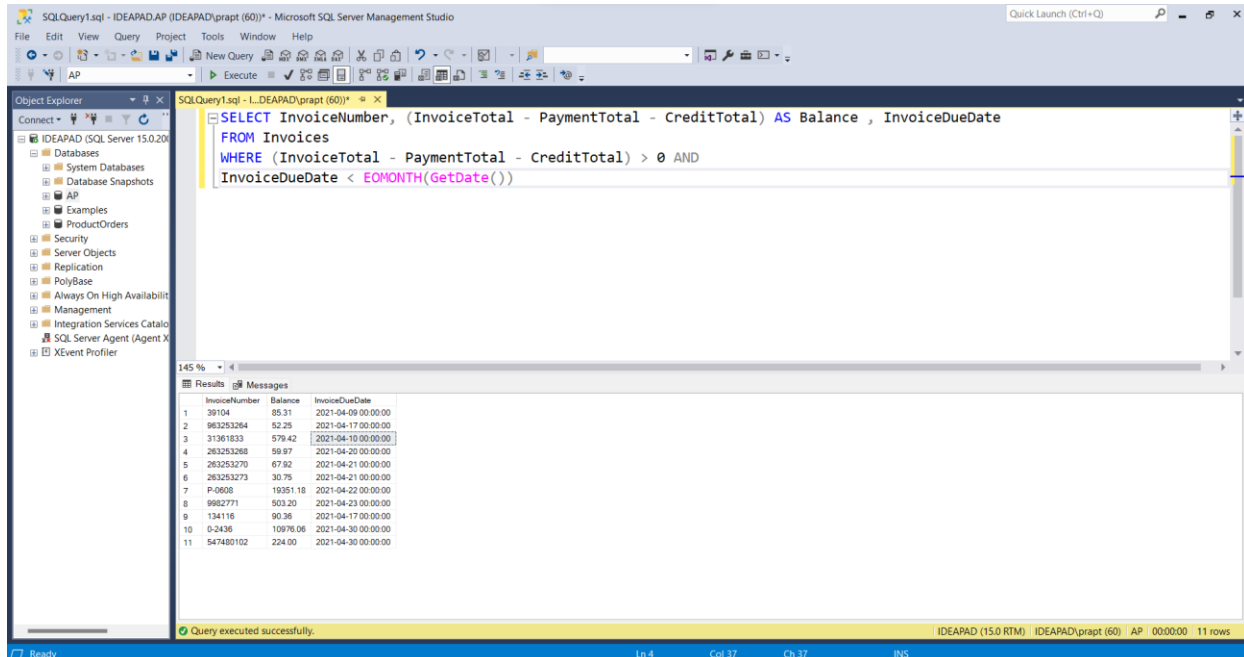
**Q3)** Modify the search expression for InvoiceDueDate from the solution for question 2. Rather than 15 days from today, return invoices due before the last day of the current month. Use AP database.

**Ans:** SELECT InvoiceNumber, (InvoiceTotal - PaymentTotal - CreditTotal) AS Balance, InvoiceDueDate

FROM Invoices

WHERE (InvoiceTotal - PaymentTotal - CreditTotal) > 0 AND

InvoiceDueDate < EOMONTH(GetDate())



**Comment:** Here, SELECT statement is used to display the InvoiceNumber column, Balance column which is the result of (InvoiceTotal – PaymentTotal – CreditTotal) and InvoiceDueDate column from the Invoices table. Records are selected by putting a condition using the WHERE clause such that the Balance is not zero i.e., greater than zero which is done using the '>' operator and the InvoiceDueDate that is before the last day of the current month. This can be done using the '<' operator and the EOMONTH() function that takes the parameter of date. GetDate() is used as a parameter to get the last day of the current month.

**Remark:** EOMONTH(date) function is used to get the last day of the month which takes a date as a parameter so that we can get the last day of that date's month.

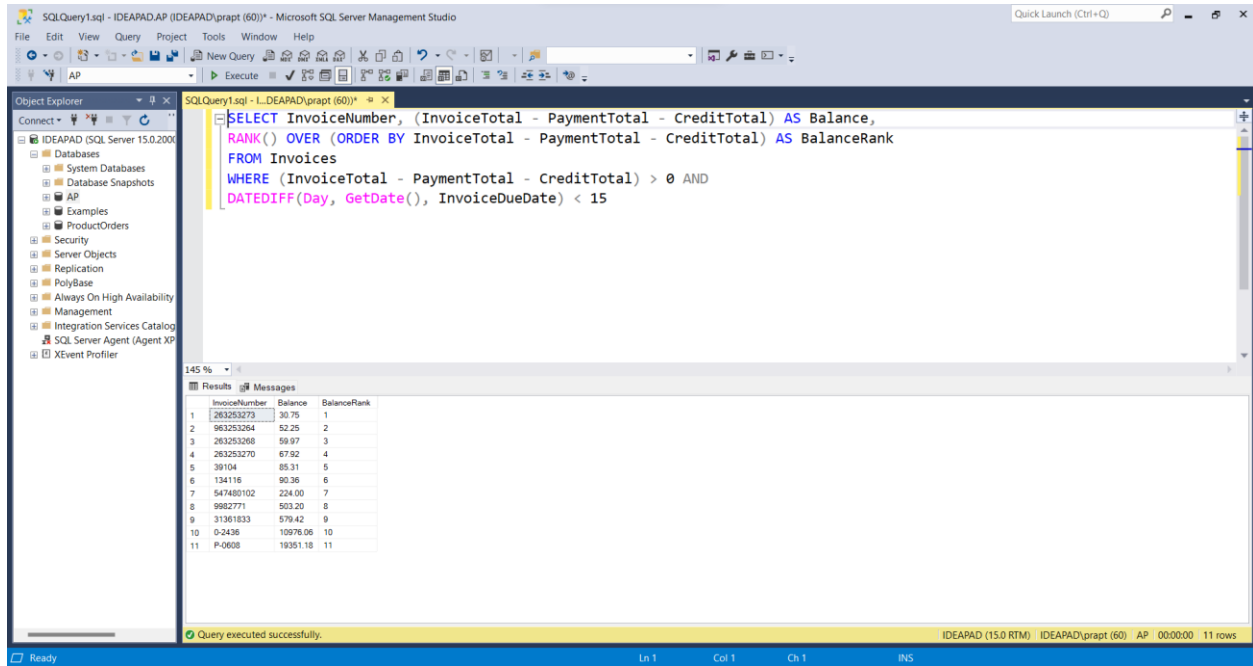
**Q4)** Add a column to the query described in question 2 that uses the RANK() function to return a column named BalanceRank that ranks the balance due in ascending order. Use AP database.

**Ans:** SELECT InvoiceNumber, (InvoiceTotal - PaymentTotal - CreditTotal) AS Balance, RANK() OVER (ORDER BY InvoiceTotal - PaymentTotal - CreditTotal) AS BalanceRank

FROM Invoices

WHERE (InvoiceTotal - PaymentTotal - CreditTotal) > 0 AND

DATEDIFF(Day, GetDate(), InvoiceDueDate) < 15



The screenshot displays the SQL Server Enterprise Manager interface. The central pane shows the execution results of a query. The query text is as follows:

```
SELECT InvoiceNumber, (InvoiceTotal - PaymentTotal - CreditTotal) AS Balance,
RANK() OVER (ORDER BY InvoiceTotal - PaymentTotal - CreditTotal) AS BalanceRank
FROM Invoices
WHERE (InvoiceTotal - PaymentTotal - CreditTotal) > 0 AND
DATEDIFF(Day, GetDate(), InvoiceDueDate) < 15
```

The results pane shows 11 rows of data:

InvoiceNumber	Balance	BalanceRank
263253273	30.75	1
963253264	52.25	2
263253268	59.97	3
263253270	67.92	4
39104	85.31	5
134116	90.36	6
547480102	224.00	7
9892771	303.20	8
31361833	579.42	9
0-2436	10976.08	10
P-0608	18951.18	11

The status bar at the bottom indicates the query was executed successfully and returned 11 rows.

**Comment:** Here, SELECT statement is used to display the InvoiceNumber column, Balance column which is the result of (InvoiceTotal – PaymentTotal – CreditTotal) and the BalanceRank column which is the result of the RANK() function on (InvoiceTotal - PaymentTotal - CreditTotal) and is used to get the rank of each row within a group of tied rows and it is sorted in the ascending order using ORDER BY.

**Remark:** RANK() function is used to rank rows tied into a partition of same rows and these same rows will be assigned a same rank.

**Remark for the lab:** Concepts related to subquery functions are used including the basic keywords, clauses and statements in this lab.