

## Lab 5: Subqueries Functions

1. Write a SELECT statement that returns two columns based on the Vendors table. The first column, Contact, is the vendor contact name in this format: Vendor Last name followed by first two letters of Vendor First name (for example, the format must look like, "Alberto Fr.") The second column, Phone, is the VendorPhone column without the area code. Only return rows for those vendors in the 209 area code. Sort the results set by first name, then last name. Use Examples database.

As you can see in the screenshot as below, here are 41 vendors set by their first two letters of first name, then last name.

The screenshot shows a SQL Server Enterprise Manager window with a query executed in the 'Examples' database. The query is as follows:

```
SELECT VendorContactLName + ' ' + LEFT (VendorContactFName, 2) + '.' AS Contact,
RIGHT(VendorPhone, 8) AS Phone
FROM Vendors
WHERE SUBSTRING ( VendorPhone, 2, 3) = 209
ORDER BY VendorContactFName, VendorContactLName;
```

The results are displayed in a table with two columns: Contact and Phone. The table shows 14 rows of data, sorted by first name then last name. The status bar at the bottom indicates 'Query executed successfully.' and '41 rows'.

|    | Contact         | Phone    |
|----|-----------------|----------|
| 1  | Alexis Al.      | 555-2993 |
| 2  | Rohansen An.    | 555-5570 |
| 3  | Braydon An.     | 555-7900 |
| 4  | Leigh Bi.       | 555-9375 |
| 5  | Jair Ca.        | 555-2420 |
| 6  | Bucket Ch.      | 555-4091 |
| 7  | Sydney De.      | 555-6621 |
| 8  | Hunter De.      | 555-1534 |
| 9  | Chaddick De.    | 555-3005 |
| 10 | Kaleigh Er.     | 555-1551 |
| 11 | Alberto Fr.     | 555-1205 |
| 12 | Finklestein Fy. | 555-7785 |
| 13 | Spivak Ha.      | 555-2770 |
| 14 | Ronaldsen Ja.   | 555-8625 |

2. 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 12 days from today (i.e. InvoiceDueDate < today's date + 12).

As you can see in the screenshot, here are 11 selected invoices.

The screenshot shows a SQL Server Enterprise Manager window with three tabs: 'Lab5Query4.sql -...PTHKBA0\lyq (65))', 'SQLQuery3.sql -...PTHKBA0\lyq (60))\*', and 'Lab5Query2.sql -...PTHKBA0\lyq (51))'. The active tab displays a SQL query:

```
SELECT InvoiceNumber, (InvoiceTotal-PaymentTotal-CreditTotal) AS Balance, InvoiceDueDate
FROM Invoices
WHERE InvoiceDueDate < GETDATE()+12
AND (InvoiceTotal-PaymentTotal-CreditTotal) <> 0;
```

Below the query editor, the 'Results' tab is selected, showing a table with 11 rows of data. The columns are 'InvoiceNumber', 'Balance', and 'InvoiceDueDate'. The status bar at the bottom indicates 'Query executed successfully.' and 'DESKTOP-PTHKBA0 (15.0 RTM) DESKTOP-PTHKBA0\lyq (51) AP 00:00:00 11 rows'.

|    | InvoiceNumber | Balance  | InvoiceDueDate      |
|----|---------------|----------|---------------------|
| 1  | 39104         | 85.31    | 2016-04-09 00:00:00 |
| 2  | 963253264     | 52.25    | 2016-04-17 00:00:00 |
| 3  | 31361833      | 579.42   | 2016-04-10 00:00:00 |
| 4  | 263253268     | 59.97    | 2016-04-20 00:00:00 |
| 5  | 263253270     | 67.92    | 2016-04-21 00:00:00 |
| 6  | 263253273     | 30.75    | 2016-04-21 00:00:00 |
| 7  | P-0608        | 19351.18 | 2016-04-22 00:00:00 |
| 8  | 9982771       | 503.20   | 2016-04-23 00:00:00 |
| 9  | 134116        | 90.36    | 2016-04-17 00:00:00 |
| 10 | 0-2436        | 10976.06 | 2016-04-30 00:00:00 |
| 11 | 547480102     | 224.00   | 2016-04-30 00:00:00 |

**3. Modify the search expression for InvoiceDueDate from the solution for question 2. Rather than 12 days from today, return invoices due before the last day of the current month.**

As the query shows, there are 11 invoices due before the last day of the current month.

The screenshot shows a SQL Server Enterprise Manager window with three tabs: Lab5Query3.sql, Lab5Query2.sql, and Lab5Query4.sql. The active tab, Lab5Query3.sql, contains the following SQL query:

```
SELECT InvoiceNumber, (InvoiceTotal-PaymentTotal-CreditTotal) AS Balance, InvoiceDueDate
FROM Invoices
WHERE InvoiceDueDate < DATEADD(DAY,-1,DATEADD(MONTH,DATEDIFF(MONTH,0,GETDATE())+1,0))
AND (InvoiceTotal-PaymentTotal-CreditTotal) <> 0;
```

Below the query editor, the 'Results' pane displays 11 rows of data. The status bar at the bottom indicates 'Query executed successfully.' and '11 rows'.

|    | InvoiceNumber | Balance  | InvoiceDueDate      |
|----|---------------|----------|---------------------|
| 1  | 39104         | 85.31    | 2016-04-09 00:00:00 |
| 2  | 963253264     | 52.25    | 2016-04-17 00:00:00 |
| 3  | 31361833      | 579.42   | 2016-04-10 00:00:00 |
| 4  | 263253268     | 59.97    | 2016-04-20 00:00:00 |
| 5  | 263253270     | 67.92    | 2016-04-21 00:00:00 |
| 6  | 263253273     | 30.75    | 2016-04-21 00:00:00 |
| 7  | P-0608        | 19351.18 | 2016-04-22 00:00:00 |
| 8  | 9982771       | 503.20   | 2016-04-23 00:00:00 |
| 9  | 134116        | 90.36    | 2016-04-17 00:00:00 |
| 10 | 0-2436        | 10976.06 | 2016-04-30 00:00:00 |
| 11 | 547480102     | 224.00   | 2016-04-30 00:00:00 |

Query executed successfully. DESKTOP-PTHKBA0 (15.0 RTM) DESKTOP-PTHKBA0\lyq (55) AP 00:00:00 11 rows

4. 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.

As the screenshot shows, you could see the ascending order of balance.

The screenshot displays a SQL Server Enterprise Manager window with two tabs: 'Lab5Query4.sql -...PTHKBA0\lyq (63)' and 'Lab5Query1.sql -...PTHKBA0\lyq (64)'. The active tab shows a SQL query that selects invoice numbers, calculated balances, due dates, and a rank based on the balance. The query is as follows:

```
SELECT InvoiceNumber, (InvoiceTotal-PaymentTotal-CreditTotal) AS Balance, InvoiceDueDate,
RANK()OVER (ORDER BY (InvoiceTotal- PaymentTotal- CreditTotal) ASC) AS BalanceRank
FROM Invoices
WHERE InvoiceDueDate < GETDATE()+12
AND (InvoiceTotal-PaymentTotal-CreditTotal) <> 0;
```

Below the query editor, the 'Results' pane shows the output of the query. The results are displayed in a table with four columns: InvoiceNumber, Balance, InvoiceDueDate, and BalanceRank. The data is sorted by Balance in ascending order. The status bar at the bottom indicates that the query was executed successfully and returned 11 rows.

|    | InvoiceNumber | Balance  | InvoiceDueDate      | BalanceRank |
|----|---------------|----------|---------------------|-------------|
| 1  | 263253273     | 30.75    | 2016-04-21 00:00:00 | 1           |
| 2  | 963253264     | 52.25    | 2016-04-17 00:00:00 | 2           |
| 3  | 263253268     | 59.97    | 2016-04-20 00:00:00 | 3           |
| 4  | 263253270     | 67.92    | 2016-04-21 00:00:00 | 4           |
| 5  | 39104         | 85.31    | 2016-04-09 00:00:00 | 5           |
| 6  | 134116        | 90.36    | 2016-04-17 00:00:00 | 6           |
| 7  | 547480102     | 224.00   | 2016-04-30 00:00:00 | 7           |
| 8  | 9982771       | 503.20   | 2016-04-23 00:00:00 | 8           |
| 9  | 31361833      | 579.42   | 2016-04-10 00:00:00 | 9           |
| 10 | 0-2436        | 10976.06 | 2016-04-30 00:00:00 | 10          |
| 11 | P-0608        | 19351.18 | 2016-04-22 00:00:00 | 11          |

Query executed successfully. DESKTOP-PTHKBA0 (15.0 RTM) DESKTOP-PTHKBA0\lyq (63) AP 00:00:00 11 rows