



Lab 3: Summary Queries, Subqueries Solution

1. Write a SELECT statement that returns two columns: VendorName and PaymentSum, where PaymentSum is the sum of the PaymentTotal column. Group the result set by VendorName. Return only 10 rows, corresponding to the 10 vendors who've been paid the least.

```
SELECT TOP 10 VendorName, SUM(PaymentTotal) AS PaymentSum
FROM Vendors JOIN Invoices
    ON Vendors.VendorID = Invoices.VendorID
GROUP BY VendorName
ORDER BY PaymentSum ASC;
```

A screenshot of the Microsoft SQL Server Management Studio (SSMS) interface. The top window shows the query editor with the following SQL code:

```
SELECT TOP 10 VendorName, SUM(PaymentTotal) AS PaymentSum
FROM Vendors JOIN Invoices
    ON Vendors.VendorID = Invoices.VendorID
GROUP BY VendorName
ORDER BY PaymentSum ASC;
```

 The bottom window shows the 'Results' tab with a table of 10 rows. The first row is highlighted. The status bar at the bottom indicates 'Query executed successfully.' and '10 rows'.

2. Write a SELECT statement that returns three columns: VendorName, InvoiceCount and InvoiceAverage. InvoiceCount is the count of the number of invoices, and InvoiceAverage is the average of the InvoiceTotal of each vendor. Group the result set by VendorName and sort the result set so that the vendor with the highest number of invoices appears first. (HINT: InvoiceAverage = Sum of the InvoiceTotal column / InvoiceCount)

```
SELECT Vendors.VendorName, COUNT(Vendors.VendorName) AS InvoiceCount,  
       SUM(InvoiceTotal)/COUNT(Vendors.VendorName) AS InvoiceAverage  
FROM Vendors JOIN Invoices  
     ON Vendors.VendorID = Invoices.VendorID  
GROUP BY VendorName  
ORDER BY InvoiceCount DESC;
```

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

```
SELECT Vendors.VendorName, COUNT(Vendors.VendorName) AS InvoiceCount,  
       SUM(InvoiceTotal)/COUNT(Vendors.VendorName) AS InvoiceAverage  
FROM Vendors JOIN Invoices  
     ON Vendors.VendorID = Invoices.VendorID  
GROUP BY VendorName  
ORDER BY InvoiceCount DESC;
```

The query results are displayed in a table with the following columns: VendorName, InvoiceCount, and InvoiceAverage. The results are sorted by InvoiceCount in descending order.

	VendorName	InvoiceCount	InvoiceAverage
1	Federal Express Corporation	47	93.1493
2	United Parcel Service	9	2575.3288
3	Zylka Design	8	867.5312
4	Pacific Bell	6	28.5016
5	Malloy Lithographing Inc	5	23978.482
6	Roadway Package System, Inc	4	10.9175
7	Blue Cross	3	188.00
8	Cardinal Business Media, Inc.	2	132.68
9	Compuserve	2	9.95
10	Data Reproductions Corp	2	10963.655

The status bar at the bottom indicates that the query was executed successfully and returned 34 rows.

3. Write a SELECT statement that returns: AccountDescription, LineItemCount, and LineItemSum. LineItemCount is the number of entries in the InvoiceLineItems table that have that AccountNo. LineItemSum is the sum of the InvoiceLineItemAmount column for that AccountNo. Filter the result set to include only those rows with LineItemCount more than 2. Group the result set by account description, and sort it by descending LineItemCount.

```
SELECT GLAccounts.AccountDescription, COUNT(*) AS LineItemCount,  
       SUM(InvoiceLineItemAmount) AS LineItemSum  
FROM GLAccounts JOIN InvoiceLineItems  
     ON GLAccounts.AccountNo = InvoiceLineItems.AccountNo  
GROUP BY GLAccounts.AccountDescription  
HAVING COUNT(*) >= 3  
ORDER BY LineItemCount DESC;
```

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

```
SELECT GLAccounts.AccountDescription, COUNT(*) AS LineItemCount,  
       SUM(InvoiceLineItemAmount) AS LineItemSum  
FROM GLAccounts JOIN InvoiceLineItems  
     ON GLAccounts.AccountNo = InvoiceLineItems.AccountNo  
GROUP BY GLAccounts.AccountDescription  
HAVING COUNT(*) >= 3  
ORDER BY LineItemCount DESC;
```

The query results are displayed in the Results window, showing 10 rows. The results are as follows:

	AccountDescription	LineItemCount	LineItemSum
1	Freight	60	27599.65
2	Book Printing Costs	8	148759.97
3	Book Production Costs	8	6175.12
4	Telephone	7	266.01
5	Direct Mail Advertising	6	3900.77
6	Books, Dues, and Subscriptions	6	5207.32
7	Computer Equipment	3	2137.05
8	Group Insurance	3	564.00
9	Office Supplies	3	175.80
10	Outside Services	3	13394.10

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

4. Write a SELECT statement that answers the following question: What is the total amount invoiced for each AccountNo? Use the WITH ROLLUP operator to include a row that gives the grand total.

```
SELECT AccountNo, SUM(InvoiceLineItemAmount) AS LineItemSum
FROM InvoiceLineItems
GROUP BY AccountNo WITH ROLLUP;
```

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

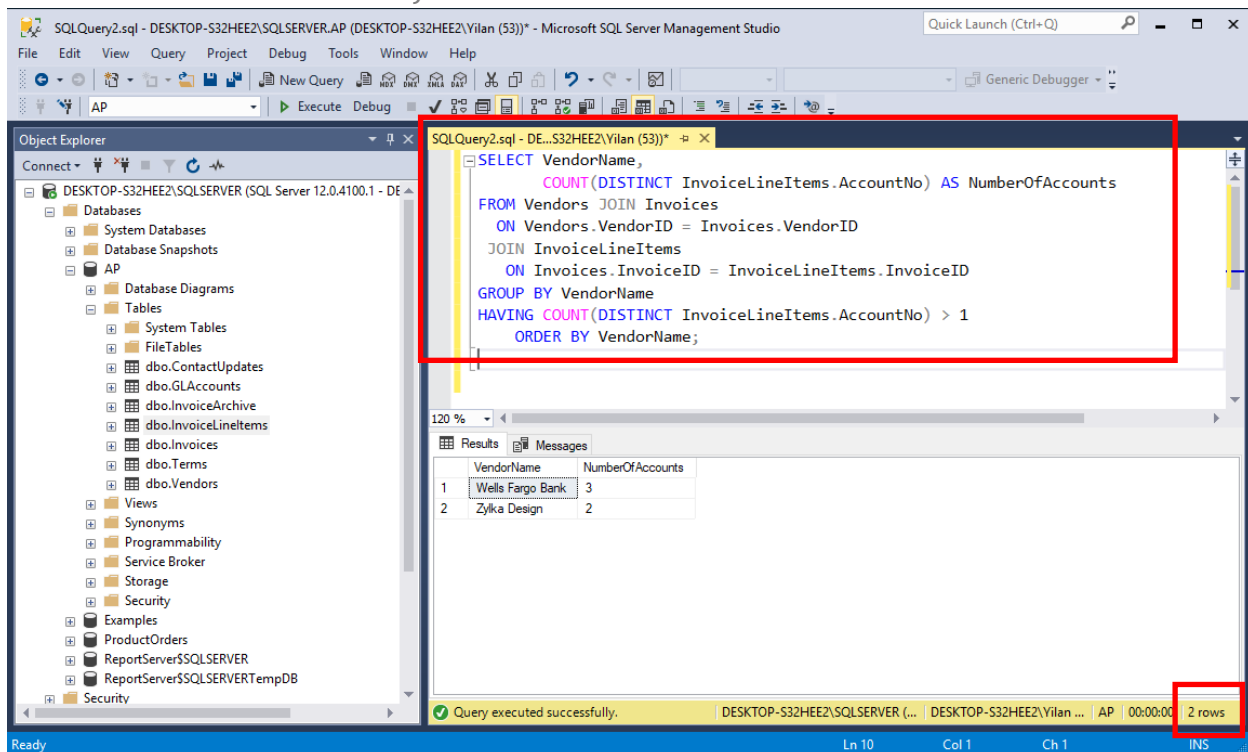
```
SELECT AccountNo, SUM(InvoiceLineItemAmount) AS LineItemSum
FROM InvoiceLineItems
GROUP BY AccountNo WITH ROLLUP;
```

The query results are displayed in a table with two columns: AccountNo and LineItemSum. The results include 21 rows of data and a final row with AccountNo NULL and LineItemSum 214290.51, representing the grand total. The status bar at the bottom indicates "Query executed successfully." and "22 rows".

AccountNo	LineItemSum
13 552	290.00
14 553	27599.65
15 570	175.80
16 572	5207.32
17 574	856.92
18 580	50.00
19 582	503.20
20 589	13394.10
21 591	220.00
22 NULL	214290.51

5. Write a SELECT statement that return the vendor name and the total number of accounts that apply to that vendor's invoices. Filter the result set to include only the vendor who is being paid more than once. (HINT: use Vendors table, Invoices table and InvoiceLineItems table)

```
SELECT VendorName,  
       COUNT(DISTINCT InvoiceLineItems.AccountNo) AS NumberOfAccounts  
FROM Vendors JOIN Invoices  
  ON Vendors.VendorID = Invoices.VendorID  
 JOIN InvoiceLineItems  
  ON Invoices.InvoiceID = InvoiceLineItems.InvoiceID  
GROUP BY VendorName  
HAVING COUNT(DISTINCT InvoiceLineItems.AccountNo) > 1  
ORDER BY VendorName;
```



6. Write a SELECT statement that returns VendorName, InvoiceNumber and InvoiceTotal for each vendor's invoice. Filter the result set to include only invoices having a PaymentTotal that is less than the average PaymentTotal for all paid invoices.

```
SELECT Vendors.VendorName, Invoices.InvoiceNumber,  
Invoices.InvoiceTotal  
FROM Invoices JOIN Vendors  
ON Vendors.VendorID = Invoices.VendorID  
AND Invoices.PaymentTotal <  
    (SELECT AVG(Invoices.PaymentTotal)  
     FROM Invoices  
     WHERE Invoices.PaymentTotal <> 0);
```

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for 'DESKTOP-S32HEE2\SQLSERVER (SQL Server 12.0.4100.1 - DE...)' with folders for Databases, System Databases, Database Snapshots, AP, Database Diagrams, Tables, System Tables, FileTables, dbo.ContactUpdates, dbo.GLAccounts, dbo.InvoiceArchive, dbo.InvoiceLineItems, dbo.Invoices, dbo.Terms, dbo.Vendors, Views, Synonyms, Programmability, Service Broker, Storage, Security, Examples, ProductOrders, ReportServer\$SQLSERVER, ReportServer\$SQLSERVERTempDB, and Security.

The SQL Query window in the center shows the following query:

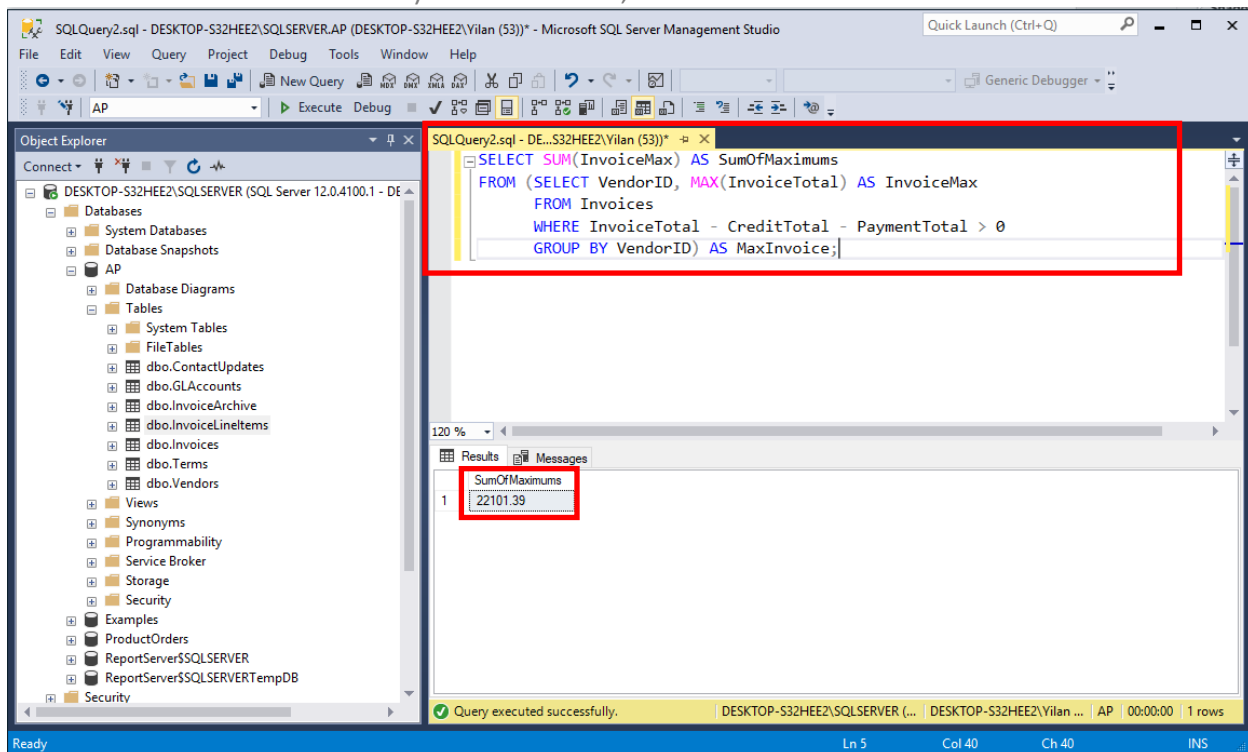
```
SELECT Vendors.VendorName, Invoices.InvoiceNumber, Invoices.InvoiceTotal  
FROM Invoices JOIN Vendors  
ON Vendors.VendorID = Invoices.VendorID  
AND Invoices.PaymentTotal <  
    (SELECT AVG(Invoices.PaymentTotal)  
     FROM Invoices  
     WHERE Invoices.PaymentTotal <> 0);
```

The Results window at the bottom displays the query output as a table with 10 rows and 3 columns: VendorName, InvoiceNumber, and InvoiceTotal. The status bar at the bottom indicates 'Query executed successfully.' and '94 rows'.

	VendorName	InvoiceNumber	InvoiceTotal
1	Federal Express Corporation	263253241	40.20
2	Federal Express Corporation	963253234	138.75
3	Federal Express Corporation	2-000-2993	144.70
4	Federal Express Corporation	963253251	15.50
5	Federal Express Corporation	963253261	42.75
6	Federal Express Corporation	963253237	172.50
7	Evans Executone Inc	125520-1	95.00
8	Zylka Design	97/488	601.95
9	Federal Express Corporation	263253250	42.67
10	Federal Express Corporation	963253262	42.50

7. Write a SELECT statement that returns the sum of the largest unpaid invoices submitted by each vendor. Use a derived table that returns MAX(InvoiceTotal) grouped by VendorID, filtering for invoices with a balance due. (HINT: Balance = InvoiceTotal – CreditTotal – PaymentTotal)

```
SELECT SUM(InvoiceMax) AS SumOfMaximums
FROM (SELECT VendorID, MAX(InvoiceTotal) AS InvoiceMax
      FROM Invoices
      WHERE InvoiceTotal - CreditTotal - PaymentTotal > 0
      GROUP BY VendorID) AS MaxInvoice;
```



8. Write a SELECT statement that returns the name, city, and state of each vendor that's located in a unique city and state. In other words, don't include vendors that have a city and state in common with another vendor. Sort the result set by VendorState in descending order, and VendorCity in ascending order.

```
SELECT VendorName, VendorCity, VendorState
FROM Vendors
WHERE VendorState + VendorCity NOT IN
(SELECT VendorState + VendorCity
FROM Vendors
GROUP BY VendorState + VendorCity
HAVING COUNT(*) > 1)
ORDER BY VendorState DESC, VendorCity ASC;
```

