

1. `SELECT TOP 2 Artist, SUM(Quantity*UnitPrice) AS 'TotalEarning'
FROM ProductOrders.dbo.Items AS I, ProductOrders.dbo.OrderDetails AS O
WHERE I.ItemID = O.ItemID
GROUP BY Artist
ORDER BY TotalEarning DESC;`

The screenshot shows the SQL Server Management Studio interface. The query window contains the following T-SQL code:

```
--Lichen Liang
--SELECT TOP 2 Artist, SUM(Quantity*UnitPrice) AS 'TotalEarning'
--FROM ProductOrders.dbo.Items AS I, ProductOrders.dbo.OrderDetails AS O
--WHERE I.ItemID = O.ItemID
--GROUP BY Artist
--ORDER BY TotalEarning DESC;
```

The results pane displays the following data:

Artist	TotalEarning
Jazz	318.10
No Rest For The Weary	279.00

Below the results, a message states "Query executed successfully."

Listing the top 2 Authors with their total earnings.

2. `SELECT AccountDescription,
COUNT(*) AS 'LineItemCount',
SUM(InvoiceLineItemAmount) AS 'LineItemSum'
FROM AP.dbo.InvoiceLineItems AS I, AP.dbo.GLAcounts AS G
WHERE I.AccountNo = G.AccountNo
GROUP BY AccountDescription
ORDER BY LineItemCount DESC;`

The screenshot shows the SQL Server Management Studio interface. The query window contains the following T-SQL code:

```
--Lichen Liang
--SELECT AccountDescription,
--COUNT(*) AS 'LineItemCount',
--SUM(InvoiceLineItemAmount) AS 'LineItemSum'
--FROM AP.dbo.InvoiceLineItems AS I, AP.dbo.GLAcounts AS G
--WHERE I.AccountNo = G.AccountNo
--GROUP BY AccountDescription
--ORDER BY LineItemCount DESC;
```

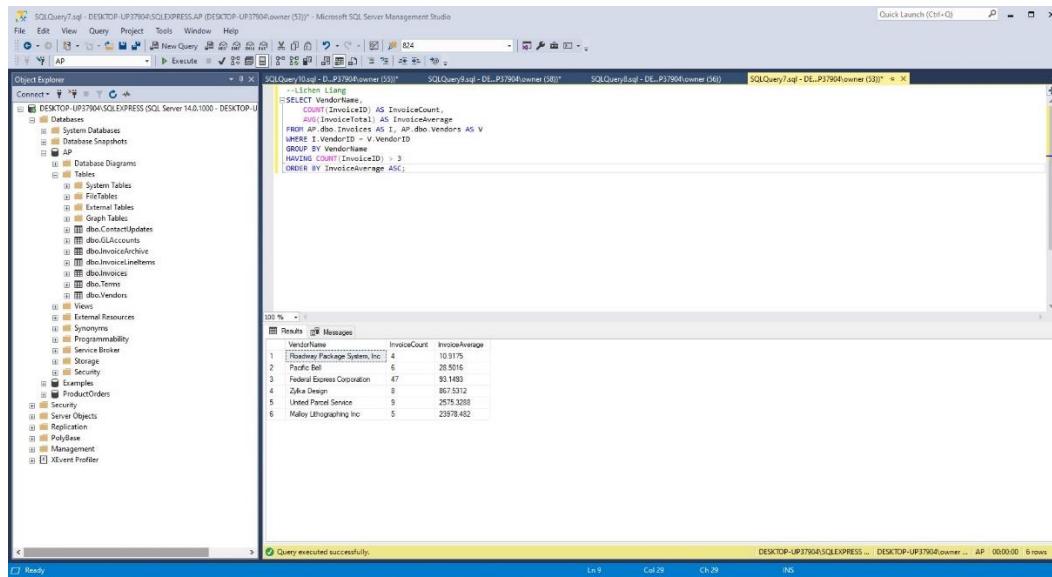
The results pane displays the following data:

AccountDescription	LineItemCount	LineItemSum
Direct Mail Advertising	6	3980.77
Outside Services	3	1394.10
Computer Equipment	3	2177.05
Group Insurance	3	554.00
Office Supplies	3	175.80
Other Equipment	1	356.48
Meals	1	50.00
Travel and Accommodation	1	501.20
UCI	1	1600.00
Utilities	1	16.62
Logistics	1	280.00
Parcels	1	175.00
Building Lease	1	1760.00
Building Maintenance	1	450.00
Business Licenses and Taxes	1	856.52
Postage	1	290.00

Below the results, a message states "Query executed successfully."

Listing account description with their line item count and the corresponding sum.

3. SELECT VendorName,
 COUNT(InvoiceID) AS InvoiceCount,
 AVG(InvoiceTotal) AS InvoiceAverage
 FROM AP.dbo.Invoices AS I, AP.dbo.Vendors AS V
 WHERE I.VendorID = V.VendorID
 GROUP BY VendorName
 HAVING COUNT(InvoiceID) > 3
 ORDER BY InvoiceAverage ASC;



The screenshot shows the SQL Server Management Studio interface with four tabs open. The main query window contains the following T-SQL code:

```

SELECT VendorName,
       COUNT(InvoiceID) AS InvoiceCount,
       AVG(InvoiceTotal) AS InvoiceAverage
  FROM AP.dbo.Invoices AS I, AP.dbo.Vendors AS V
 WHERE I.VendorID = V.VendorID
 GROUP BY VendorName
 HAVING COUNT(InvoiceID) > 3
 ORDER BY InvoiceAverage ASC;
  
```

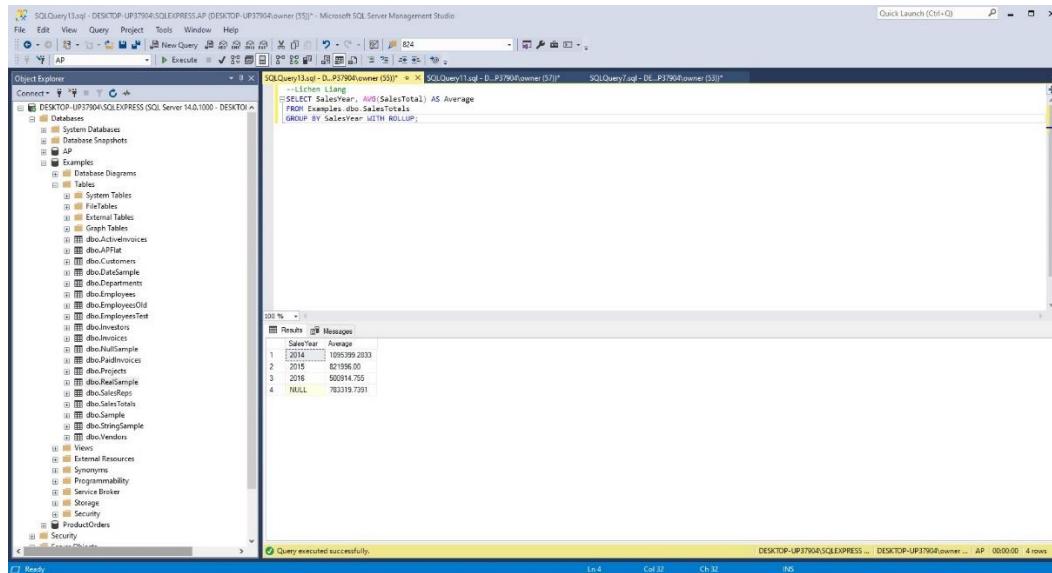
The results pane displays a table with six rows, showing vendor names, invoice counts, and average invoice totals. The data is as follows:

VendorName	InvoiceCount	InvoiceAverage
Boeing Package Systems, Inc.	4	10.9175
Pacific Bell	6	28.5016
Federal Express Corporation	47	93.1493
Zylex Design	8	867.5312
United Parcel Service	9	2975.308
Kathy UltraGraphing Inc	5	2379.482

The status bar at the bottom indicates "Query executed successfully." and "6 rows".

Getting the vendors with their invoice count and the average, with count more than three

4. SELECT SalesYear, AVG(SalesTotal) AS Average
 FROM Examples.dbo.SalesTotals
 GROUP BY SalesYear WITH ROLLUP;



The screenshot shows the SQL Server Management Studio interface with four tabs open. The main query window contains the following T-SQL code:

```

SELECT SalesYear, AVG(SalesTotal) AS Average
  FROM Examples.dbo.SalesTotals
 GROUP BY SalesYear WITH ROLLUP;
  
```

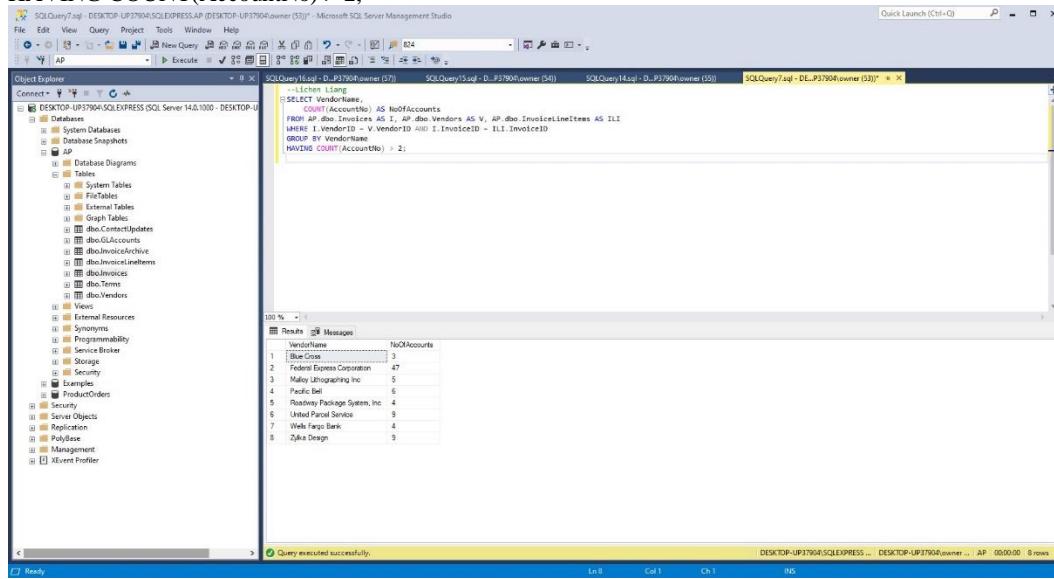
The results pane displays a table with four rows, showing sales years and their average sales totals. The data is as follows:

SalesYear	Average
2014	1095399.28
2015	821996.00
2016	500914.755
NULL	783319.7391

The status bar at the bottom indicates "Query executed successfully." and "4 rows".

The average for 2014, 2015, 2016 are 1095399.28, 821996, 500914.755 respectfully with grand average of 783319.7391

5. SELECT VendorName,
 COUNT(AccountNo) AS NoOfAccounts
 FROM AP.dbo.Invoices AS I,
 AP.dbo.Vendors AS V,
 AP.dbo.InvoiceLineItems AS ILI
 WHERE I.VendorID = V.VendorID AND I.InvoiceID = ILI.InvoiceID
 GROUP BY VendorName
 HAVING COUNT(AccountNo) > 2;



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'DESKTOP-UP37904\SQLExpress\AP'. The 'Results' tab on the right displays the results of the following query:

```

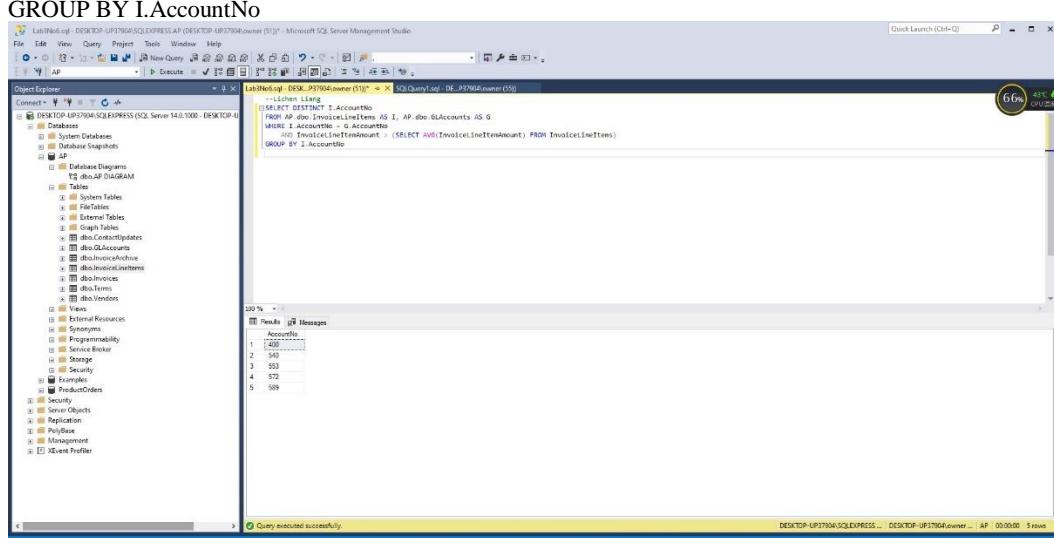
SELECT VendorName,
       COUNT(AccountNo) AS NoOfAccounts
  FROM AP.dbo.Invoices AS I, AP.dbo.Vendors AS V, AP.dbo.InvoiceLineItems AS ILI
 WHERE I.VendorID = V.VendorID AND I.InvoiceID = ILI.InvoiceID
 GROUP BY VendorName
 HAVING COUNT(AccountNo) > 2;
  
```

The results table shows the following data:

VendorName	NoOfAccounts
AT&T Corp.	47
Federal Express Corporation	47
Major Utegraphics Inc	5
Pacific Bell	6
Roadway Package System, Inc	4
United Parcel Service	9
Wells Fargo Bank	4
Zjuka Design	9

Vendors with more than 2 or more accounts.

6. SELECT DISTINCT I.AccountNo
 FROM AP.dbo.InvoiceLineItems AS I, AP.dbo.GLAcounts AS G
 WHERE I.AccountNo = G.AccountNo
 AND InvoiceLineItemAmount >
 (SELECT AVG(InvoiceLineItemAmount)
 FROM InvoiceLineItems)
 GROUP BY I.AccountNo



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'DESKTOP-UP37904\SQLExpress\AP'. The 'Results' tab on the right displays the results of the following query:

```

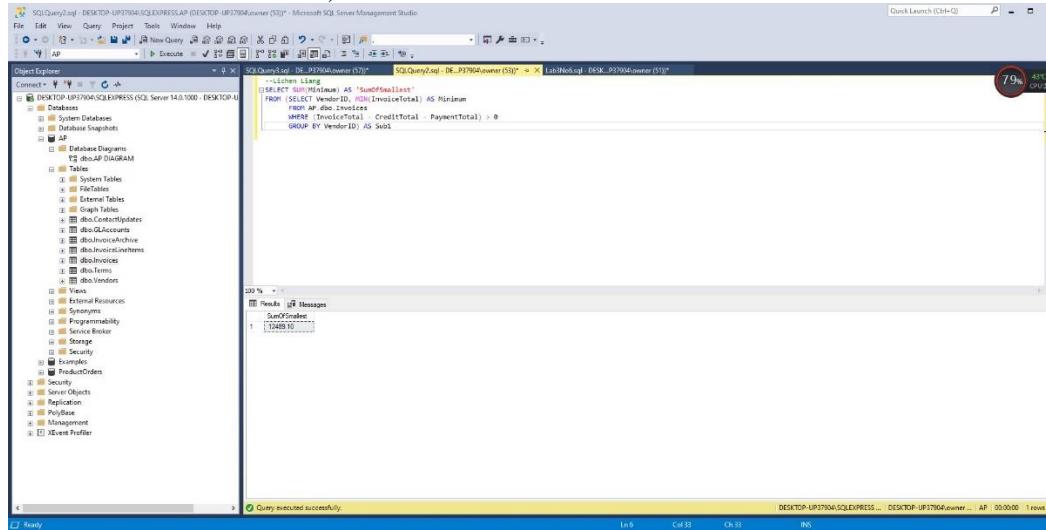
SELECT DISTINCT I.AccountNo
  FROM AP.dbo.InvoiceLineItems AS I, AP.dbo.GLAcounts AS G
 WHERE I.AccountNo = G.AccountNo
 AND InvoiceLineItemAmount >
 (SELECT AVG(InvoiceLineItemAmount) FROM InvoiceLineItems)
 GROUP BY I.AccountNo
  
```

The results table shows the following data:

AccountNo
409
540
573
572
589

Account numbers that have Invoice Line Item Amount greater than the average of all Invoice Line Item Amount

7. `SELECT SUM(Minimum) AS 'SumOfSmallest'
 FROM (SELECT VendorID, MIN(InvoiceTotal) AS Minimum
 FROM AP.dbo.Invoices
 WHERE (InvoiceTotal - CreditTotal - PaymentTotal) > 0
 GROUP BY VendorID) AS Sub1`



The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'AP'. The central pane contains the following T-SQL code:

```
SELECT SUM(Minimum) AS 'SumOfSmallest'
FROM (SELECT VendorID, MIN(InvoiceTotal) AS Minimum
      FROM AP.dbo.Invoices
      WHERE (InvoiceTotal - CreditTotal - PaymentTotal) > 0
      GROUP BY VendorID) AS Sub1
```

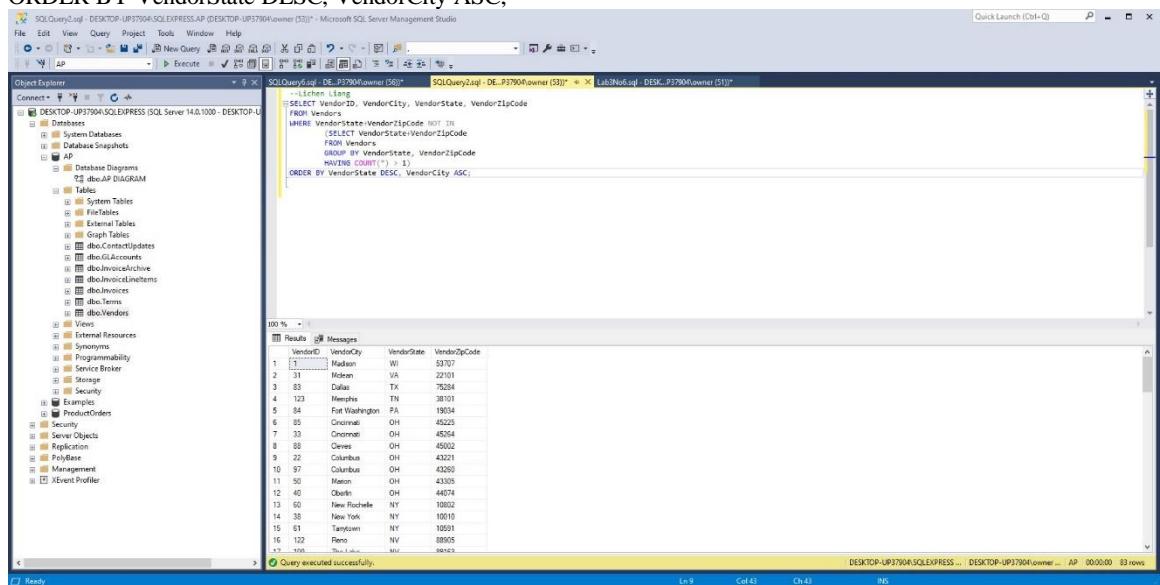
The results pane shows the output of the query:

SumOfSmallest
12489.10

A message at the bottom of the results pane says 'Query executed successfully.'

The sum of the smallest unpaid invoices.

8. `SELECT VendorID, VendorCity, VendorState, VendorZipCode
 FROM Vendors
 WHERE VendorState+VendorZipCode NOT IN
 (SELECT VendorState+Vendor ZipCode
 FROM Vendors
 GROUP BY VendorState, VendorZipCode
 HAVING COUNT(*) > 1)
 ORDER BY VendorState DESC, VendorCity ASC;`



The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'AP'. The central pane contains the following T-SQL code:

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

The results pane shows the output of the query:

VendorID	VendorCity	VendorState	VendorZipCode	
1	Madison	WI	53707	
2	Moline	VA	22101	
3	Dallas	TX	75284	
4	Henderson	NV	89014	
5	Seattle	Washington	98104	
6	Cincinnati	OH	45225	
7	Chromwell	OH	45054	
8	Orives	OH	45002	
9	22	Columbus	OH	43221
10	57	Columbus	OH	43201
11	50	Or	43205	
12	40	Ovind	OH	44674
13	60	New Rochelle	NY	10002
14	38	New York	NY	10010
15	61	Tanyon	NY	10501
16	122	Reno	NV	89505
17	100	Or	98143	

A message at the bottom of the results pane says 'Query executed successfully.'

Vendors that do not share the same state and zip with another vendor

Remarks

In this lab we practiced query and subqueries. We learned how to use GROUP BY, HAVING along with multiple aggregate functions, and subqueries that are necessary for a comparison. Using the knowledge that we learned from class, we set up conditions for a query that meets the lab requirement. I think this lab is very efficient practice for lectures as they are somewhat similar. The challenge would be more specific requirements and more complicated queries.