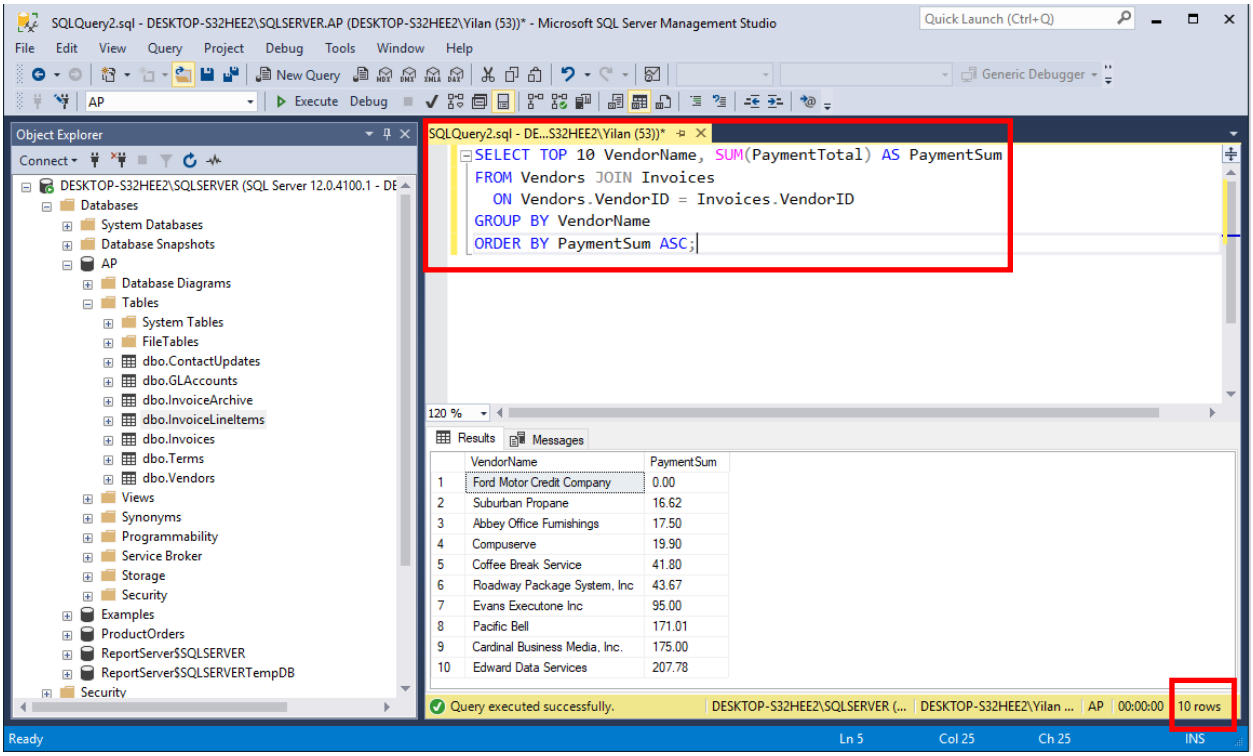
Lab 3: Summary Queries, Subqueries Solution

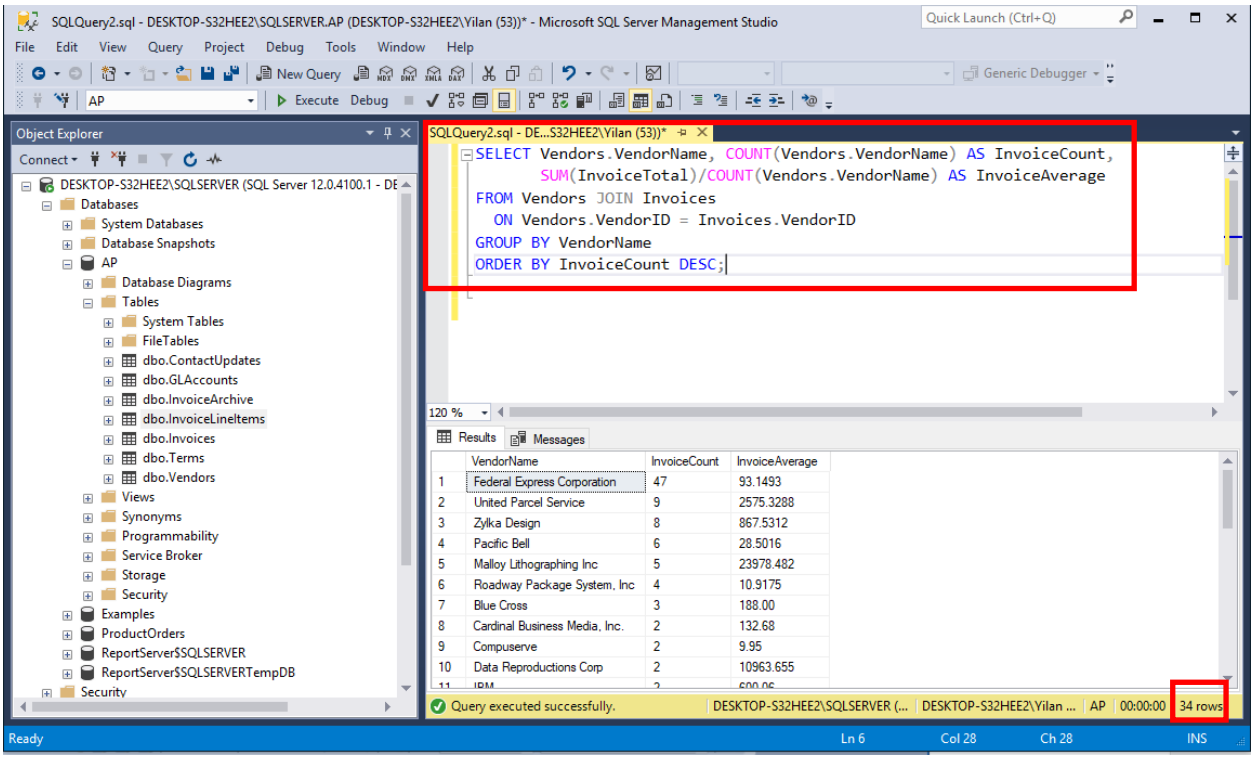
1. Write a SELECT statement that returns two columns: VendorName and PaymentSum, where PaymetSum is the sum of the PaymentTotal column. Group the result set by VendorName. Return only 10 rows, corresponding to the 10 vendors who've been paid the least.

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



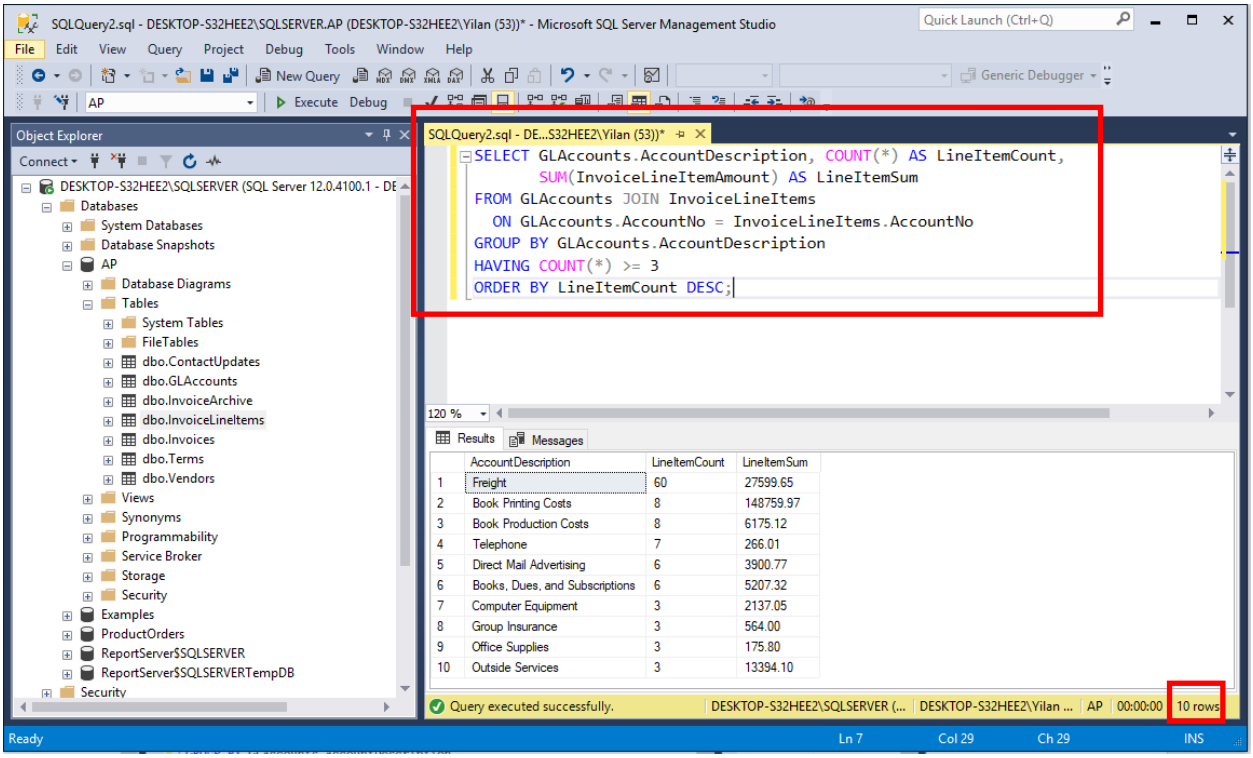
2. Write a SELECT statement that returns three columns: VendorName, InvoiceCount and InvoiceAverage. InvoiceCount is the count of the number of invoices, and InvoiceAverage is the average of the InvoiceTotal of each vendor. Group the result set by VendorName and sort the result set so that the vendor 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;
```



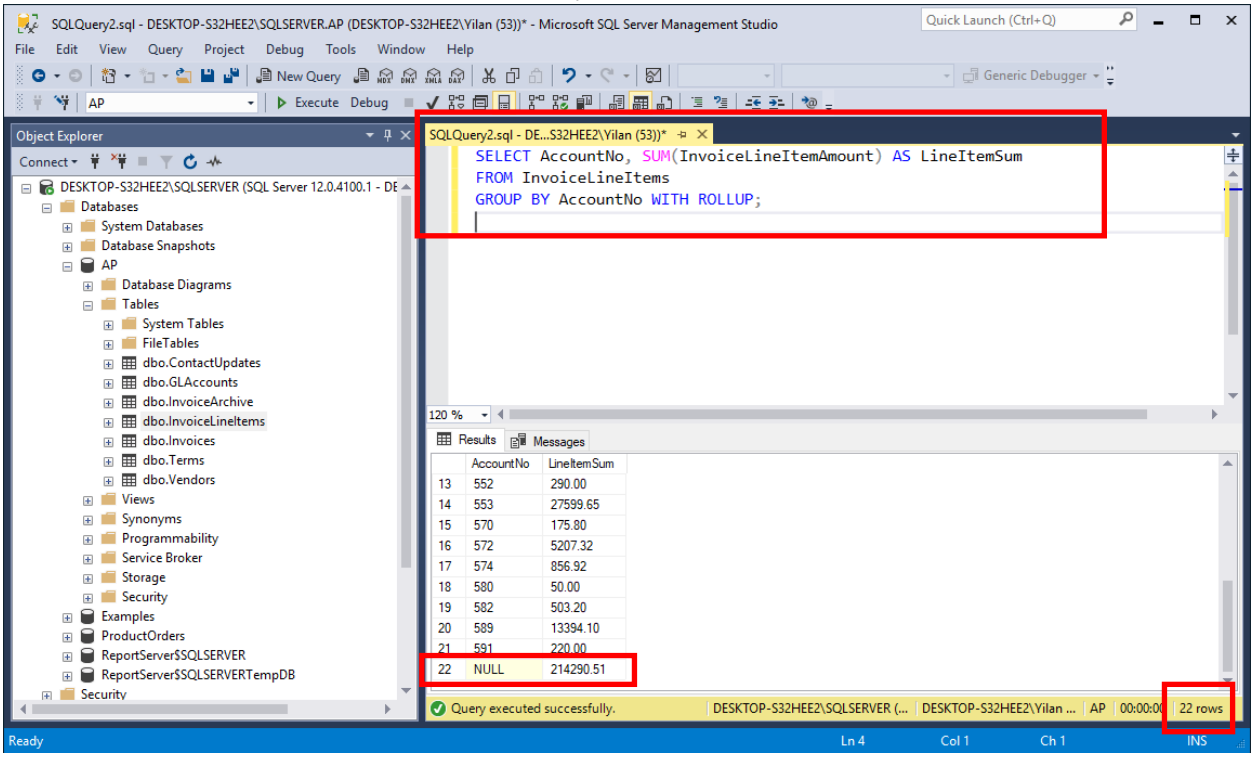
3. Write a SELECT statement that returns: AccountDescription, LineItemCount, and LineItemSum. LineItemCount is the number of entries in the InvoiceLineItems table that have that AccountNo. LineItemSum is the sum of the InvoiceLineItemAmount column for that AccountNo. Filter the result set to include 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;
```



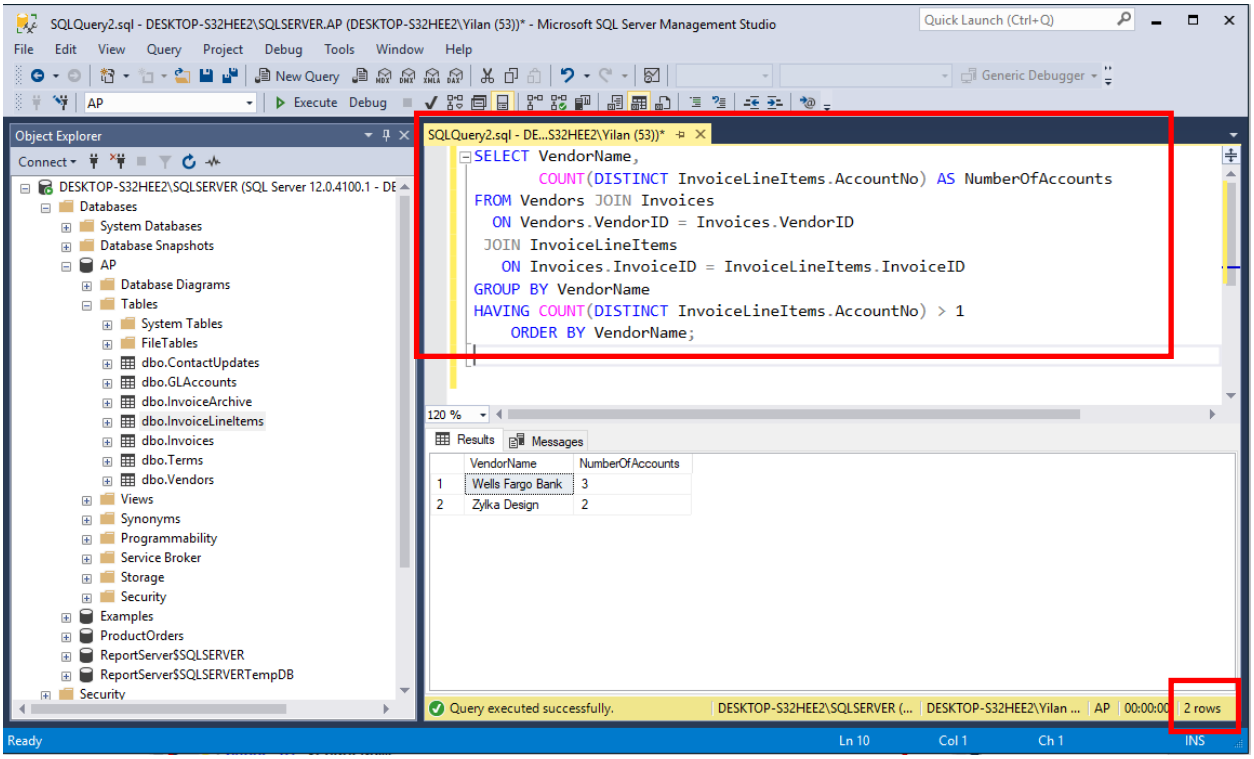
4. Write a SELECT statement that answers the following question: What is the total amount invoiced for each AccountNo? Use the WITH ROLLUP operator to include a row that gives the grand total.

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



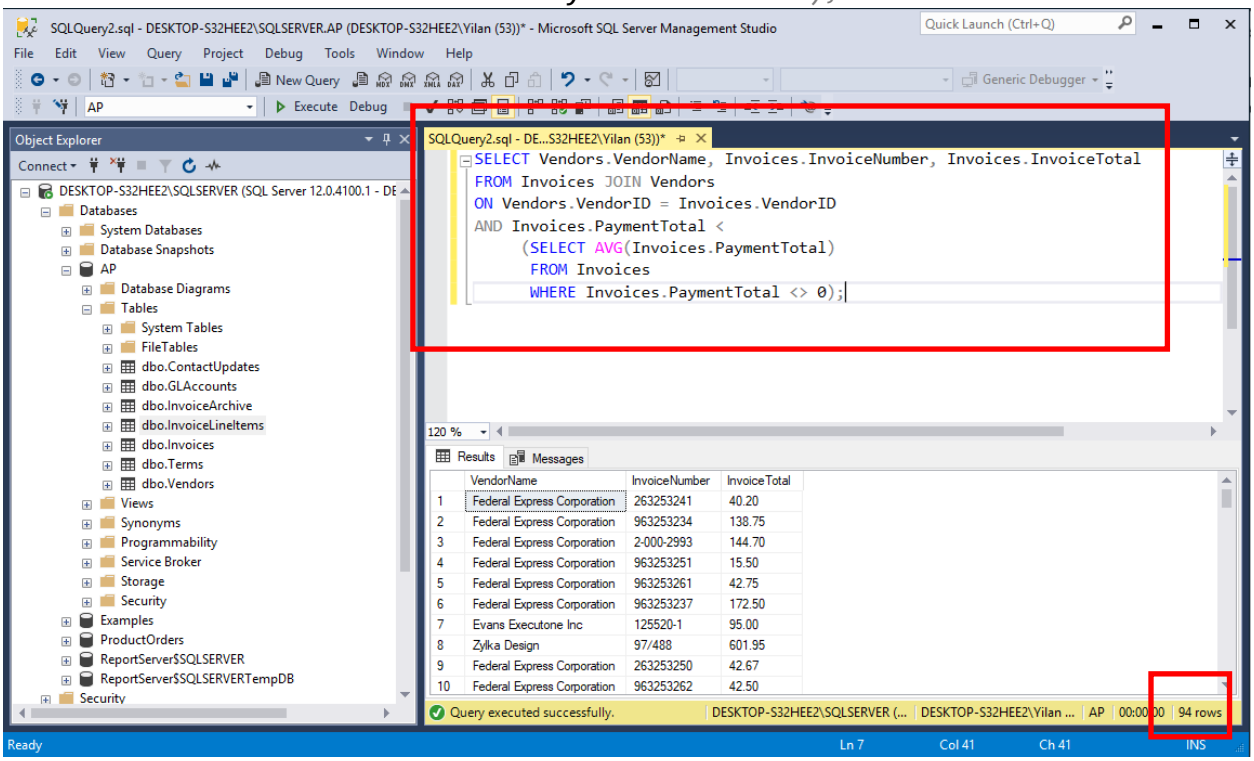
5. Write a SELECT statement that return the vendor name and the total number of accounts that apply to that vendor's invoices. Filter the result set to include only the vendor who is being paid more than once. (HINT: use Vendors table, Invoices table and InvoiceLineItems table)

```
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;
```



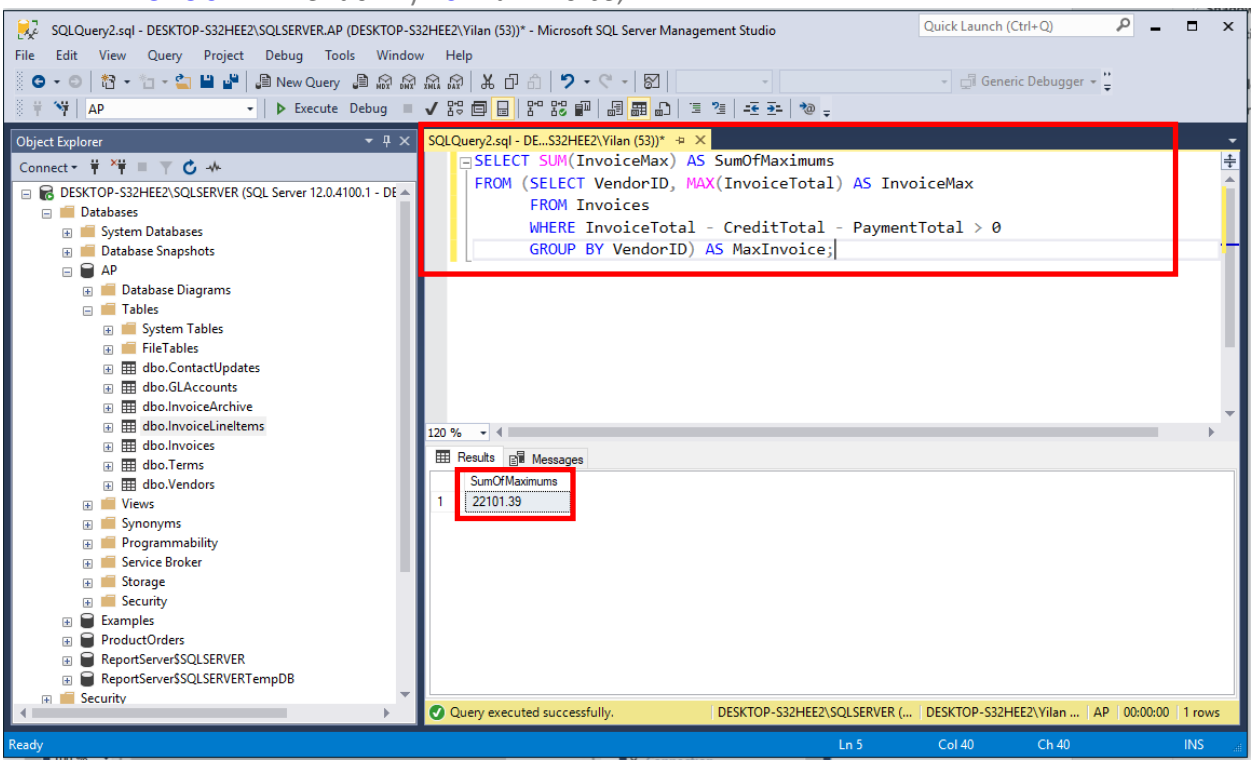
6. Write a SELECT statement that returns VendorName, InvoiceNumber and InvoiceTotal for each vendor's invoice. Filter the result set to include only invoices having a PaymentTotal that is less than the average PaymentTotal for all paid invoices.

```
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);
```



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 VendroState 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;
```