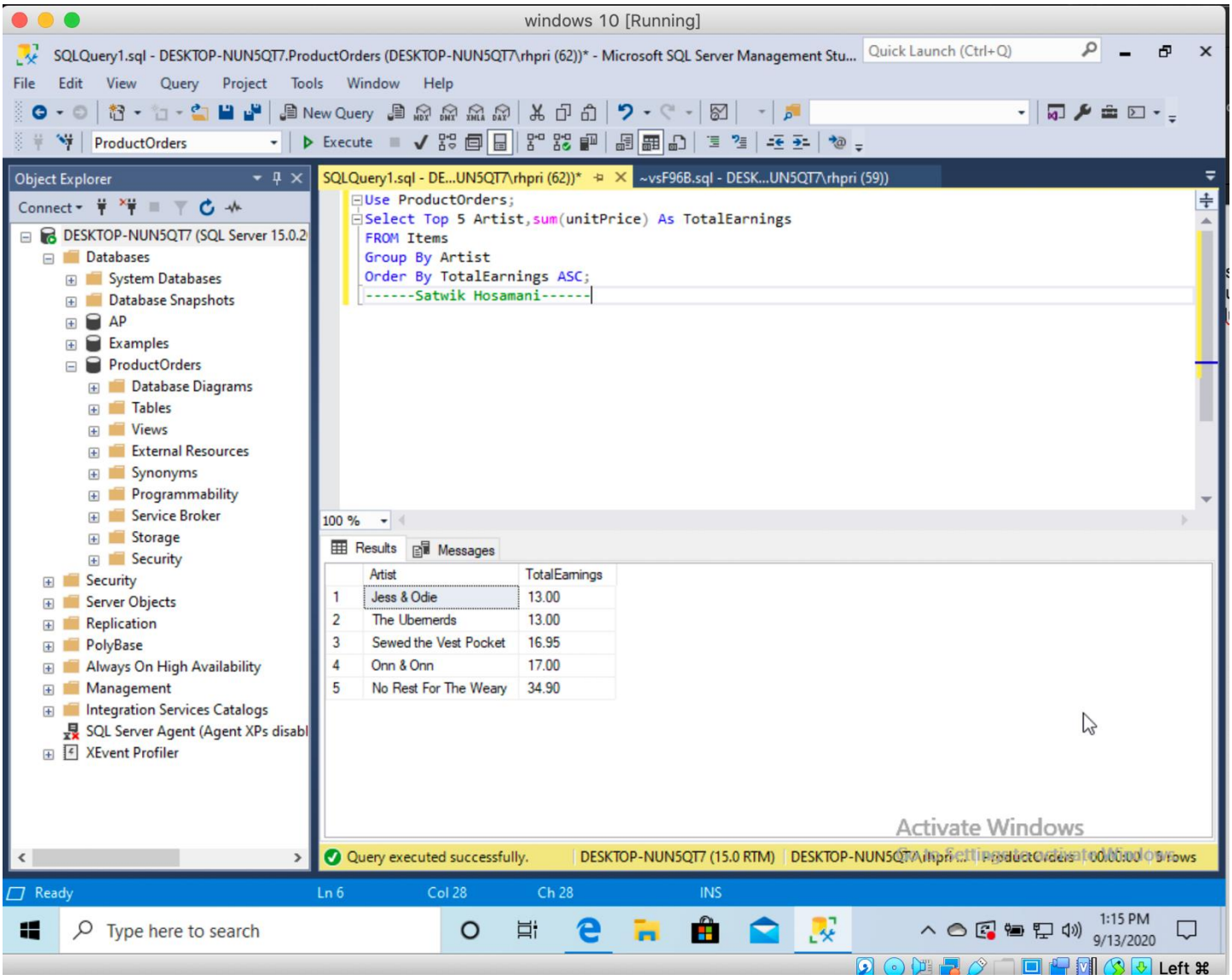
Q-1) Write a SELECT statement that returns two columns: Artist and their individual TotalEarnings, where TotalEarnings is the sum of the UnitPrice column. Return 5 artists who've earned the least in ascending order of their earnings. Use ProductOrders database.

SOL:

```
Use ProductOrders;
Select Top 5 Artist,sum(unitPrice) As TotalEarnings
FROM Items
Group By Artist
Order By TotalEarnings ASC;
------Satwik Hosamani------
```

Comments:The query uses top function to take the first n number of rows which satisfy the condition .To arrange the rows in the ascending order order by asc is used.

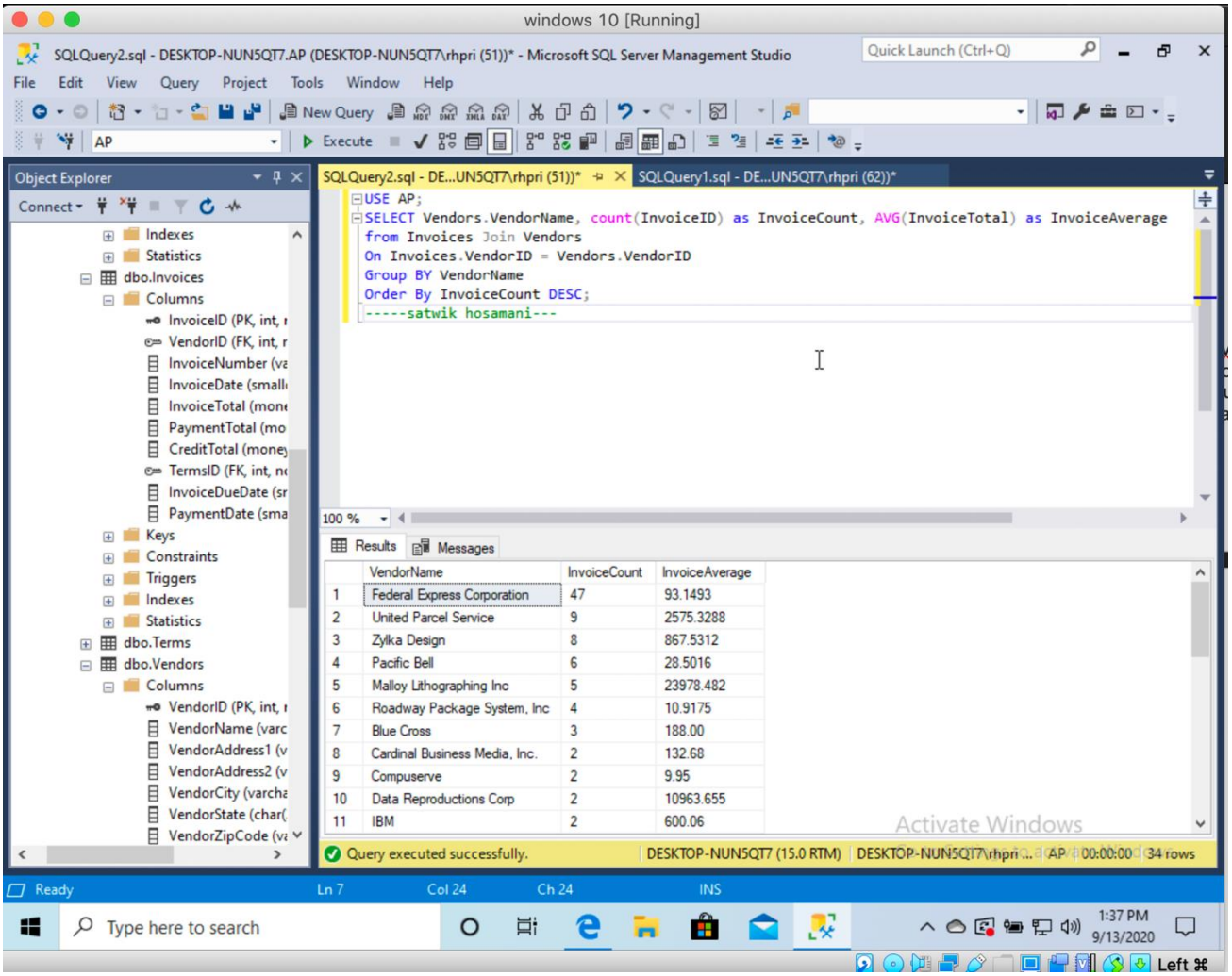


Q-2) Write a SELECT statement that returns three columns: VendorName, InvoiceCount and InvoiceAverage. InvoiceCount is the count of the number of invoices, and InvoiceAverage is the average of the InvoiceTotal of each vendor. Group the result set by VendorName and sort the result in descending order of number of invoices. Use AP database.

SOL:

```
USE AP;
SELECT Vendors.VendorName, count(InvoiceID) as InvoiceCount, AVG(InvoiceTotal) as
InvoiceAverage
from Invoices Join Vendors
On Invoices.VendorID = Vendors.VendorID
Group BY VendorName
Order By InvoiceCount DESC;
-----satwik hosamani---
```

Comments:The count aggregate function is used to find the count of the Invoices using InvoiceID also average aggregate function is used to find the average of the InvoiceTotal.To arrange the vendorName together GroupBy is used and for descending order OrderBy Desc

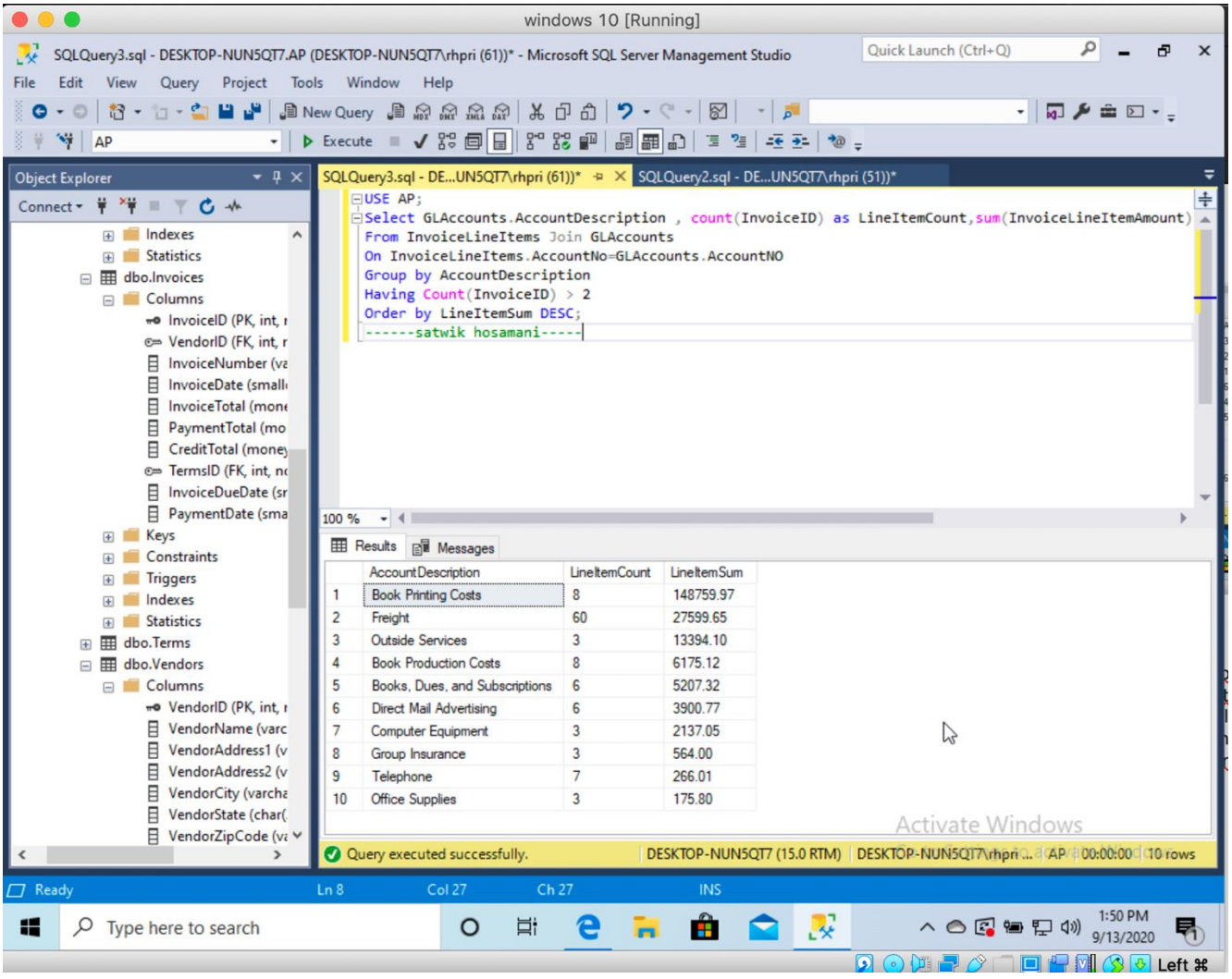


3) Write a SELECT statement that returns: AccountDescription, LineItemCount, and LineItemSum. LineItemCount is the number of entries in the InvoiceLineItems table that have that AccountNo. LineItemSum is the sum of the InvoiceLineItemAmount column for that AccountNo. Filter the result set to include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

SOL:

```
Use AP;
Select GLAccounts.AccountDescription , count(InvoiceID) as LineItemCount,sum(InvoiceLineItemAmount) as LineItemSum
From InvoiceLineItems Join GLAccounts
On InvoiceLineItems.AccountNo=GLAccounts.AccountNO
Group by AccountDescription
Having Count(InvoiceID) >2
Order by LineItemCount DESC;
```

Comments: The sum of the total amount of items is found using aggregate function sum which takes the column name as input and return it's total sum.To filter out the rows having clause is used because of group by and not where clause.To arrange In descending order "Orderby desc"

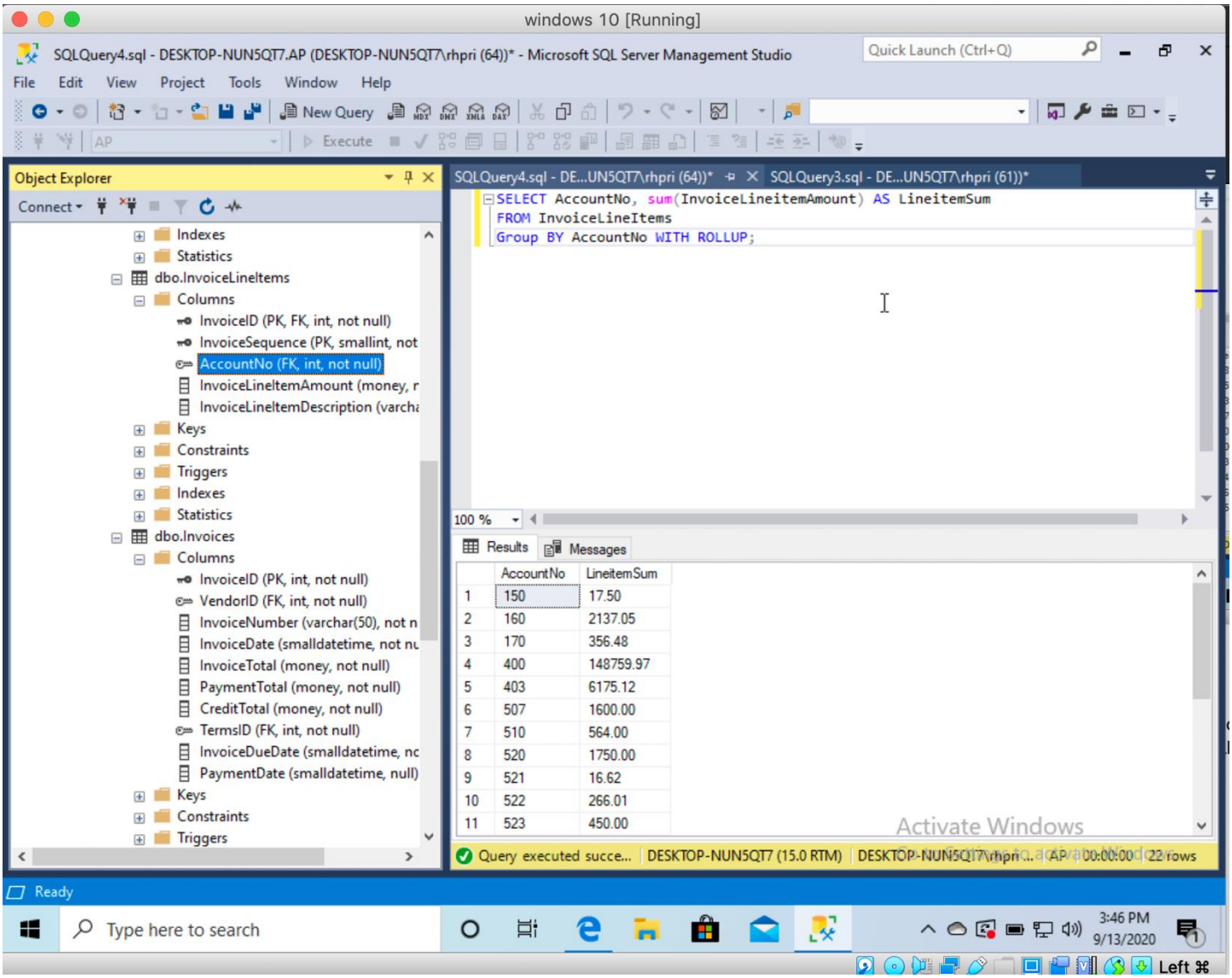


4) Write a SELECT statement that answers the following question: What is the total amount invoiced for each AccountNo? Use the WITH ROLLUP operator to include a row that gives the grand total.

SOL:

```
SELECT AccountNo, sum(InvoiceLineitemAmount) AS LineitemSum
FROM InvoiceLineItems
Group BY AccountNo WITH ROLLUP;
```

Comments: **ROLLUP** is an extension of the GROUP BY clause. It allows one to include sub totals.Here it is used to include the rows which give out the sum of amount in the accounts.

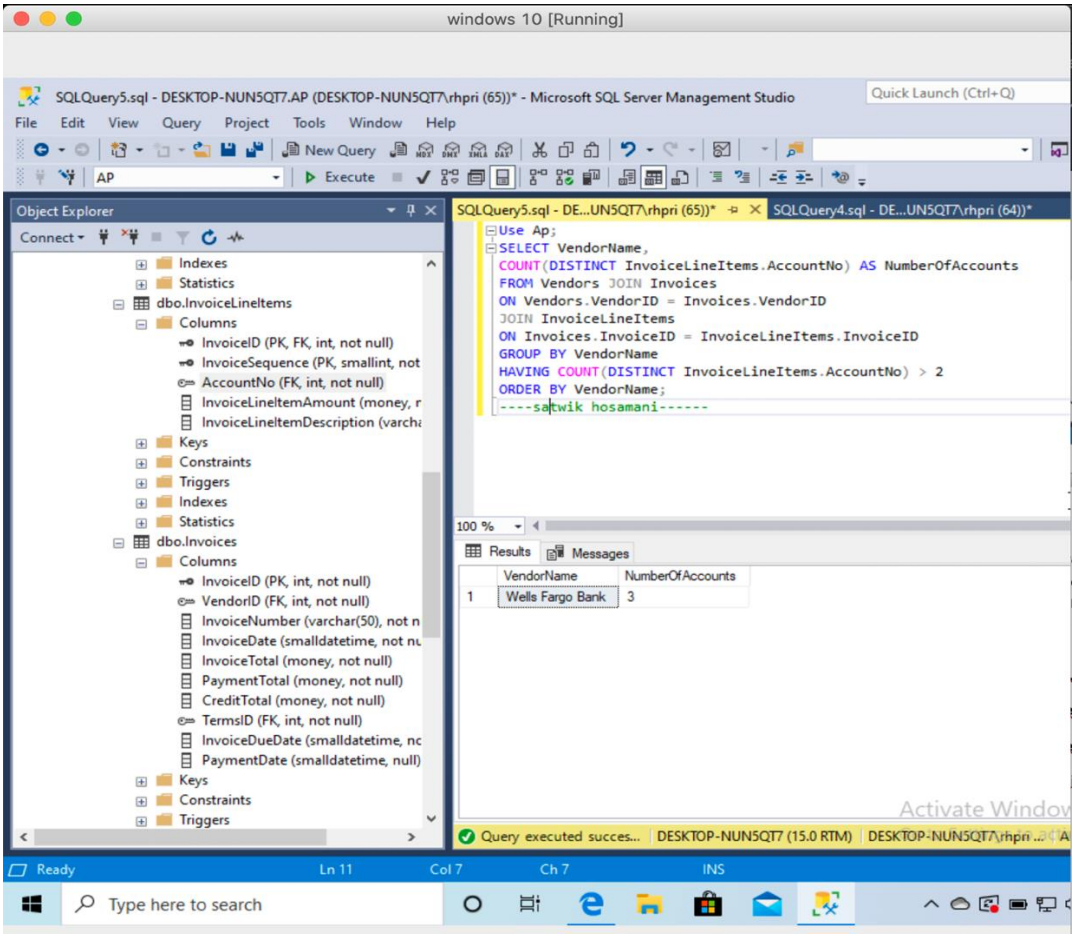


5) Write a SELECT statement that return the vendor name and the total number of accounts that apply to that vendor's invoices. Filter the result set to include only the vendor who is being paid more than twice. (HINT: use Vendors table, Invoices table and InvoiceLineItems table).

SOL:

```
Use Ap;
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) > 2
ORDER BY VendorName;
```

Comments:I have to take columns from multiple takes hence I have used join on the tables. Aggregate count function is used to count the number of accounts using Account ID. Having clause is used to put the condition because of group by clause.

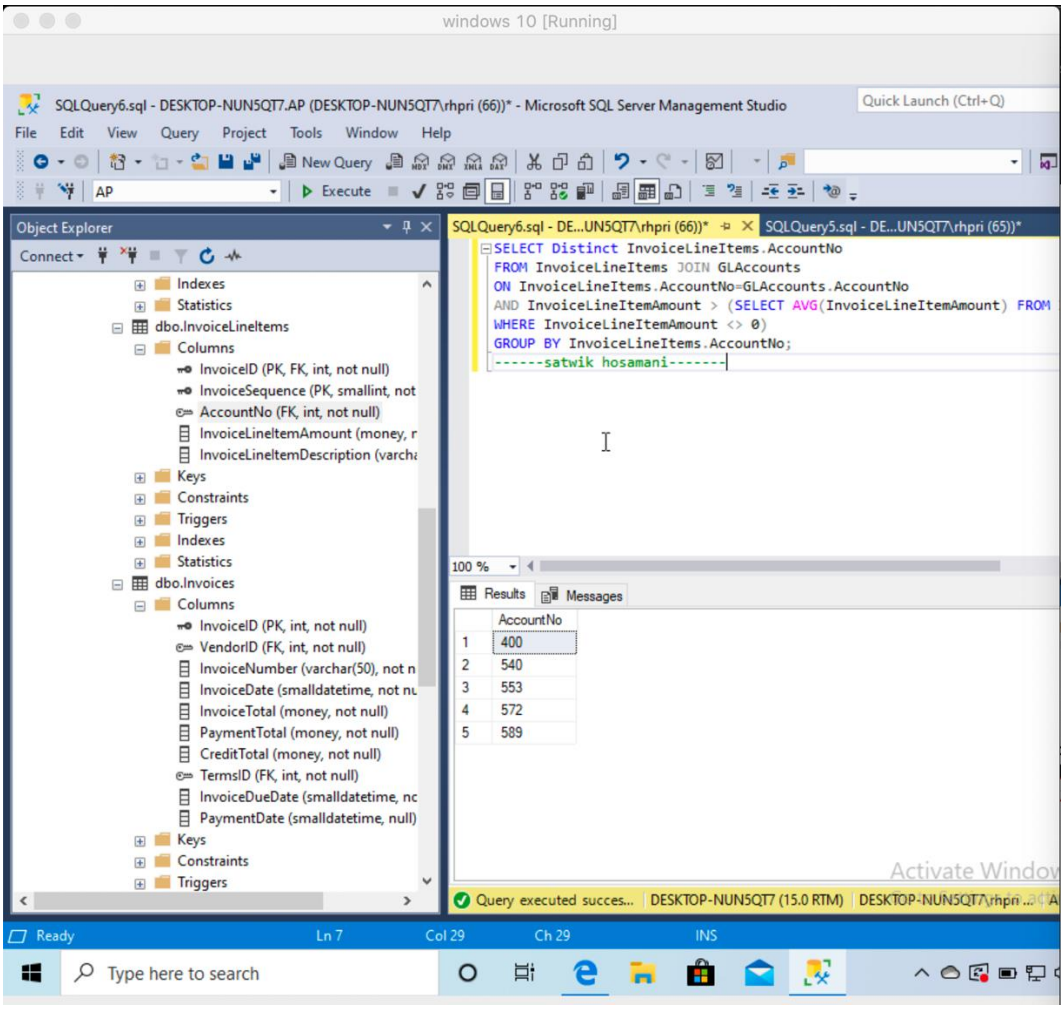


6) Write a SELECT statement that returns the distinct AccountNo for invoicelineitems (AccountNo should not be repeated in the result). Filter the result set to include only accounts having an InvoiceLineItemAmount that is greater than the average InvoiceLineItemAmount for all invoicelineitems. Use GLAccounts and InvoiceLineItems tables from AP database.

SOL:

```
SELECT Distinct InvoiceLineItems.AccountNo
FROM InvoiceLineItems JOIN GLAccounts
ON InvoiceLineItems.AccountNo=GLAccounts.AccountNo
AND InvoiceLineItemAmount > (SELECT AVG(InvoiceLineItemAmount) FROM InvoiceLineItems
WHERE InvoiceLineItemAmount <> 0)
GROUP BY InvoiceLineItems.AccountNo;
```

Comments: To provide the distinct account numbers I have used the DISTINCT operation and as I have taken columns from multiple tables hence I have used join on the tables. Condition has been checked for InvoiceLineAmount being greater than the average InvoiceLineItemAmount for all invoicelineitems. AVG function is used for calculating the average.

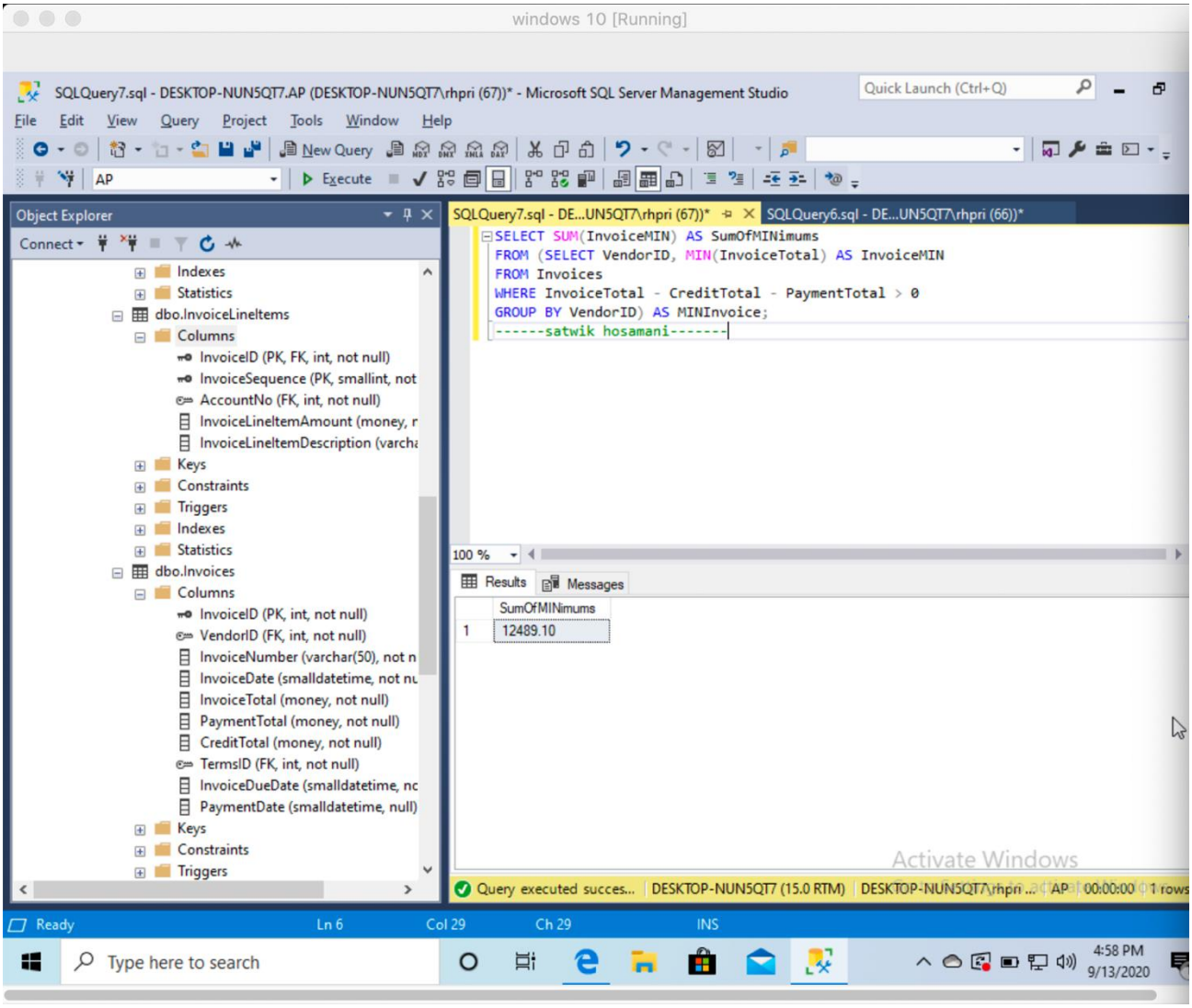


7. Write a SELECT statement that returns the sum of the smallest unpaid invoices submitted by each vendor. Use a derived table that returns MIN(InvoiceTotal) grouped by VendorID, filtering for invoices with a balance due. (HINT: Balance = InvoiceTotal – CreditTotal - PaymentTotal)

SOL:

```
USE AP;
SELECT SUM(InvoiceMIN) AS SumOfMINimums
FROM (SELECT VendorID, MIN(InvoiceTotal) AS InvoiceMIN
FROM Invoices
WHERE InvoiceTotal - CreditTotal - PaymentTotal > 0
GROUP BY VendorID) AS MINInvoice;
```

Comments: SUM function has been used to calculate the smallest unpaid invoices. A from (subquery) is used to call derive a table that returns MIN(InvoiceTotal) grouped by VendorID. MIN Function is used to calculate the minimum of invoicetotal. Balance is calculated as Balance = InvoiceTotal – CreditTotal – PaymentTotal.

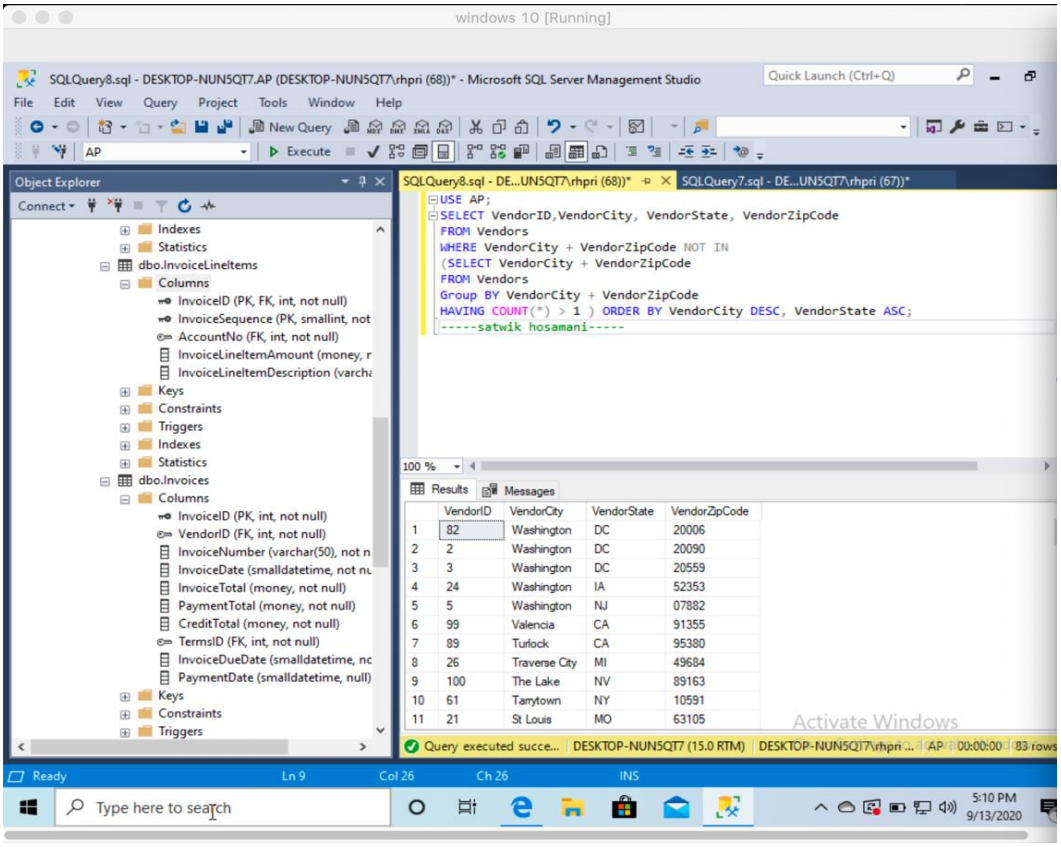


8. Write a SELECT statement that returns the id, city, state, and zip-code of each vendor that's located in a unique city with a unique zip-code (combination is unique). In other words, don't include vendors that have a city and zip-code in common with another vendor. Sort the result set by city in descending order, and state in ascending order.

SOL:

```
USE AP;
SELECT VendorID,VendorCity, VendorState, VendorZipCode
FROM Vendors
WHERE VendorCity + VendorZipCode NOT IN
(SELECT VendorCity + VendorZipCode
FROM Vendors
Group BY VendorCity + VendorZipCode
HAVING COUNT(*) > 1 ) ORDER BY VendorCity DESC, VendorState ASC;
-----satwik hosamani-----
```

Comments: To provide unique city and unique zip – code initially a subquery is derived which provides a table which returns list of VendorCity and VendorZipCode whose count is more than 1 and and in the main select statement VendorCity and VendorZipCode are filtered using the not in function. COUNT function is used to calculate the number.



Remarks: In this lab, I have studied concepts like summary queries and subqueries like HAVING, WHERE, WITH ROLL UP etc. Also, I learned the usage of aggregate functions to calculate arithmetic values.In conclusion, this lab was difficult but we learned usage of subquery and summary queries.I learnt that I need to work on my group by clause queries.