CSE 581 : INTRODUCTION TO DATABASE MANAGEMENT SYSTEMS

LAB : 3

DATE : 9/21/2022

Q1) Write a SELECT statement that returns two columns: VendorName and their individual PaymentAverage, where PaymentAverage is the average of the PaymentTotal column. Return 5 vendors who have been paid the most. ***Use AP database.***

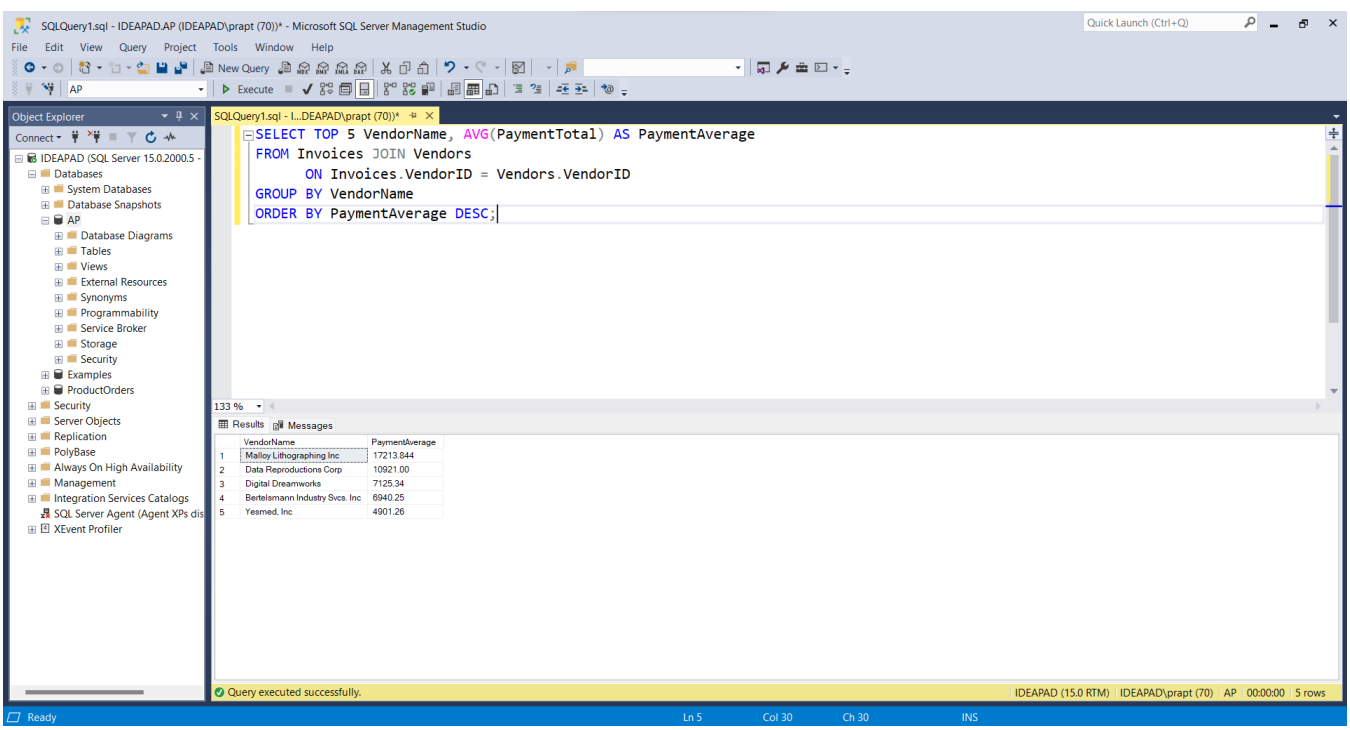
Ans: SELECT TOP 5 VendorName, AVG(PaymentTotal) AS PaymentAverage

FROM Invoices JOIN Vendors

ON Invoices.VendorID = Vendors.VendorID

GROUP BY VendorName

ORDER BY PaymentAverage DESC;



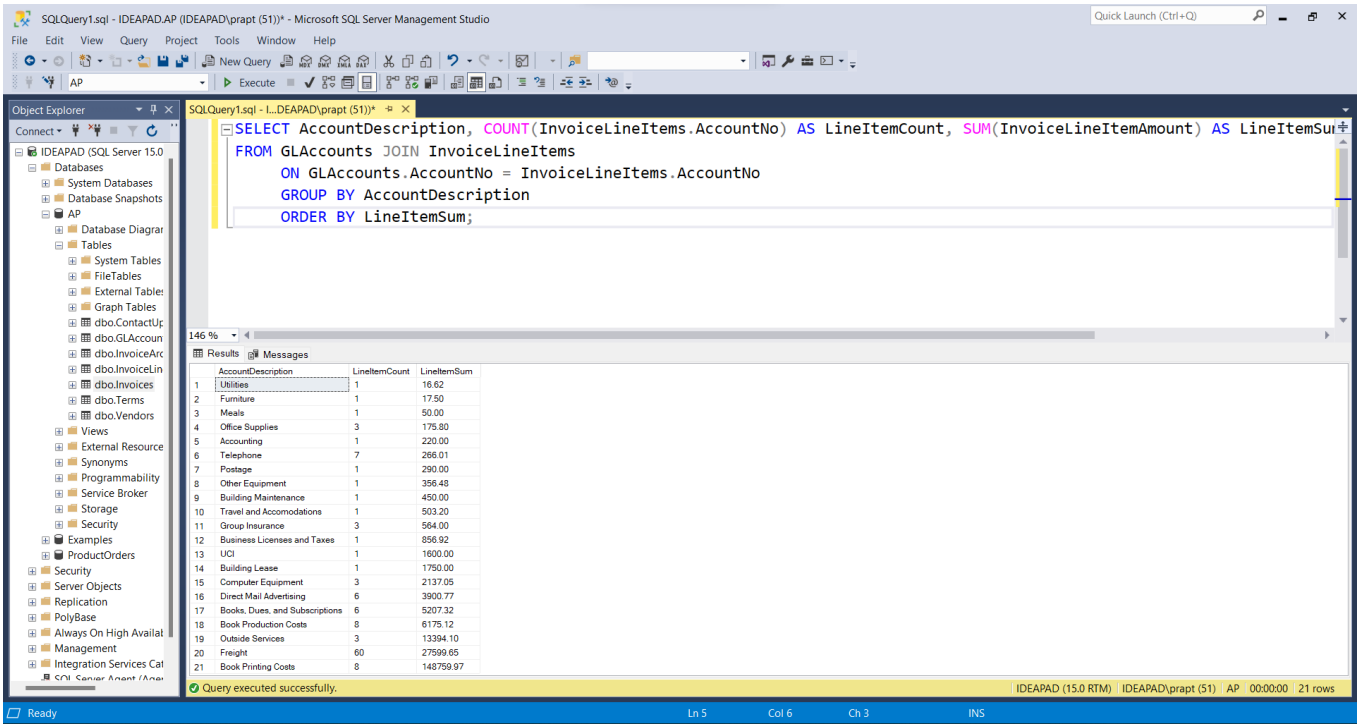
Comment: Here, SELECT statement is used to display the VendorName column and their individual PaymentAverage. The PaymentAverage column is obtained using the aggregate function AVG() that is used to calculate the average value of numeric values of the PaymentTotal column. TOP keyword is used to specify the number of rows to display from the top of the table out of several rows of the table and that number is specified next to it. JOIN is used to join the Invoices and Vendors table ON keyword is used to specify the column VendorID based on which the join is performed. The result set is grouped by VendorName using GROUP BY and is sorted in descending order to give the top 5 vendors who have been paid the most.

Remark: To display only some given number of records out of a pool of records, TOP keyword is used with the specification of the number of records to be displayed from top of the table. AVG() is an aggregate function that gives the average of the values of the column that has numeric values. GROUP BY statement is used to group the rows with same values into summary rows.

Q2) Write a SELECT statement that returns: AccountDescription, LineItemCount, and LineItemSum. LineItemCount is the number of entries in the InvoiceLineItems table that has that AccountNo. LineItemSum is the sum of the InvoiceLineItemAmount column for that AccountNo. Group the result set by account description, and sort it in ascending order of LineItemSum. ***Use AP database.***

Ans: SELECT AccountDescription, COUNT(InvoiceLineItems.AccountNo) AS LineItemCount,

```
    SUM(InvoiceLineItemAmount) AS LineItemSum
    FROM GLAccounts JOIN InvoiceLineItems
        ON GLAccounts.AccountNo = InvoiceLineItems.AccountNo
    GROUP BY AccountDescription
    ORDER BY LineItemSum;
```



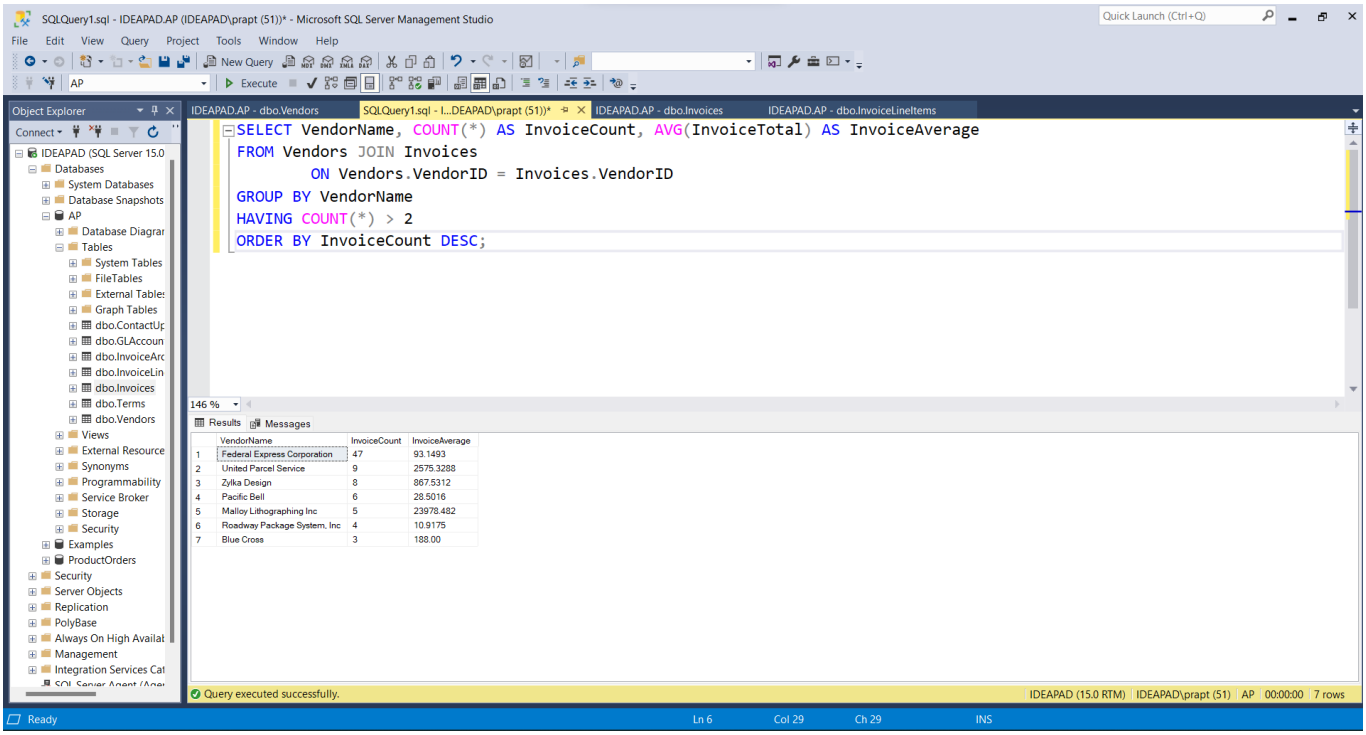
Comment: Here, SELECT statement is used to display the AccountDescription, LineItemCount and LineItemSum columns by joining the GLAccounts and InvoiceLineItems tables on AccountNo using JOIN and ON keywords. LineItemCount column is obtained by performing the aggregate COUNT() function on the AccountNo column. LineItemSum column is obtained by performing the aggregate SUM() function on the InvoiceLineItemSum column. The result set is grouped by AccountDescription column and it is sorted by the LineItemSum column in ascending order.

Remark: Aggregate functions like COUNT() and SUM() when used must be accompanied by GROUP BY clause and HAVING clause.

Q3) 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. Filter the result set to include only those rows with InvoiceCount more than 2. Group the result set by VendorName and sort the result set in descending order of InvoicesCount. ***Use AP database.***

Ans: SELECT VendorName, COUNT(*) AS InvoiceCount, AVG(InvoiceTotal) AS

```
InvoiceAverage
FROM Vendors JOIN Invoices
        ON Vendors.VendorID = Invoices.VendorID
GROUP BY VendorName
HAVING COUNT(*) > 2
ORDER BY InvoiceCount DESC;
```



Comment: Here, SELECT statement is used to return VendorName, InvoiceCount and InvoiceAverage columns. InvoiceCount is obtained by applying the aggregate function COUNT(*) to the table to get the number of invoices. InvoiceAverage column is obtained by applying the aggregate function AVG() to InvoiceTotal column. JOIN is performed on the tables Vendors and Invoices on VendorID and the result set is grouped by VendorName using GROUP BY statement to get summary rows of the result set. HAVING clause is used to put conditions to the rows obtained after grouping the rows by VendorName whose InvoiceCount is more than 2. Finally, the result set is sorted in descending order by InvoiceCount.

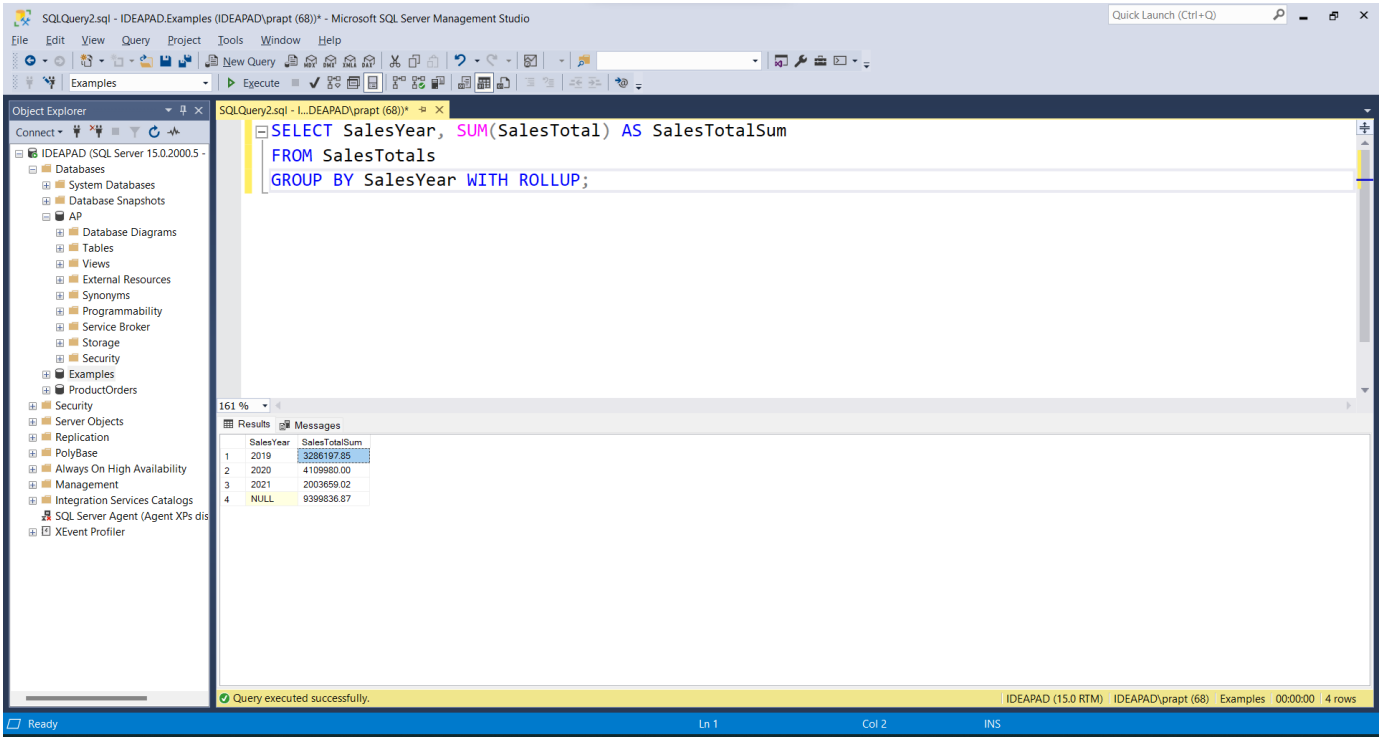
Remark: HAVING clause is used instead of WHERE clause when conditions are to be placed on the grouped rows and when aggregate functions are used.

Q4) Write a SELECT statement that answers the following question: What is the sum of sales for each "Sale Year"? Use the WITH ROLLUP operator to include a row that gives the grand sum. *Use SalesTotals table from* ***Examples database.***

Ans: SELECT SalesYear, SUM(SalesTotal) AS SalesTotalSum

FROM SalesTotals

GROUP BY SalesYear WITH ROLLUP;



Comment: Here, SELECT statement is used to display the SalesYear and SalesTotalSum columns. SalesTotalSum column is obtained by applying the aggregate function SUM() to the SalesTotal column that adds the sales of same years. The result set is grouped by the GROUP BY statement on SalesYear. WITH ROLLUP clause is used with the GROUP BY statement to obtain the grand total of the rows which is displayed in the rows which is added at the end of the result set.

Remark: WITH ROLLUP clause adds one more row to the result set to display the grand total of the total that is obtained and summarized by the grouping operator.

Q5) Write a SELECT statement that returns the vendor's 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. Sort the result set in ascending order of VendorName. *(HINT: Use Vendors table, Invoices table and InvoiceLineItems table of AP database).*

Ans: SELECT VendorName, COUNT(Distinct(InvoiceLineItems.AccountNo))

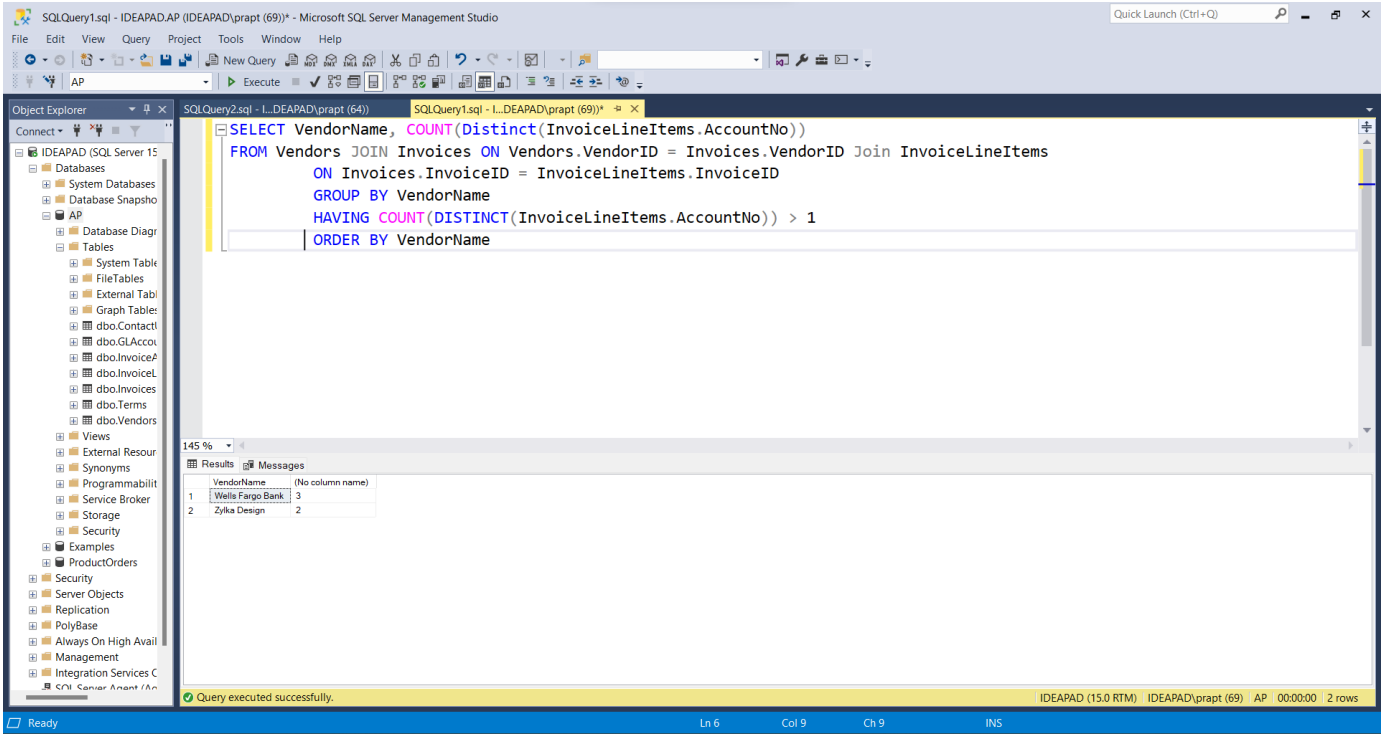
FROM Vendors JOIN Invoices ON Vendors.VendorID = Invoices.VendorID Join InvoiceLineItems

ON Invoices.InvoiceID = InvoiceLineItems.InvoiceID

GROUP BY VendorName

HAVING COUNT(DISTINCT(InvoiceLineItems.AccountNo)) > 1

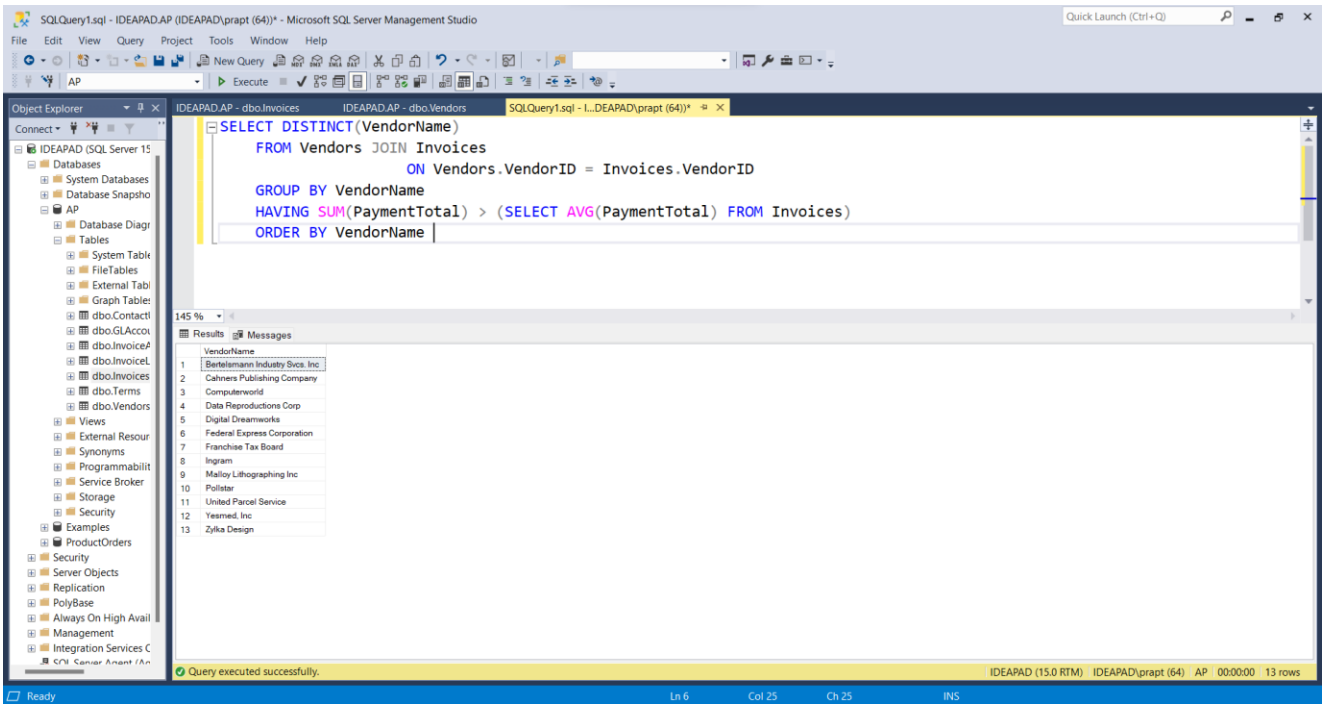
ORDER BY VendorName



Comment: Here, SELECT statement is used to display the VendorName column and the count of the number of account numbers associated with that vendor's invoices. JOIN keyword is used to join the Vendors and Invoices tables on VendorID using ON keyword and Invoices and InvoiceLineItems table on InvoiceID using ON keyword. Using GROUP BY keyword, group the result set by VendorName to form a summary. HAVING is used to apply condition here instead of WHERE clause because WHERE cannot be used with aggregate function and grouping operators. Condition is applied to display only those vendors who are paid more than once using DISTINCT keyword to avoid duplicate values and using COUNT() function to count the vendors. '>' operator is used to check the vendors that are paid more than once. Finally, the result set is sorted by VendorName in ascending order.

Q6) Write a SELECT statement that returns the distinct VendorName (i.e. VendorName should not be repeated in the result). Filter the result set to include only those vendors with invoices having a PaymentTotal that is greater than the average PaymentTotal for all invoices. Sort the result set in ascending order of VendorName. ***Use AP database.***

Ans:
```
SELECT DISTINCT(VendorName)
    FROM Vendors JOIN Invoices
        ON Vendors.VendorID = Invoices.VendorID
    GROUP BY VendorName
    HAVING SUM(PaymentTotal) > (SELECT AVG(PaymentTotal) FROM Invoices)
    ORDER BY VendorName
```



Comment: Here, SELECT statement is used to display the VendorName column with non-repeatable values for which DISTINCT keyword is used. JOIN keyword is used to join the two tables Vendors and Invoices on VendorID using ON keyword. Grouping is done using GROUP BY at VendorName column. A condition is applied to check whether the sum pf payment total of the Vendors is greater than the average of the payment total. For average PaymentTotal a subquery is written in which AVG() function is used to find the average of the PaymentTotal column from the Invoices table and that subquery is compared to the sum of the PaymentTotal column of the outer query. This, gives the result set with the distinct vendors that have PaymentTotal greater than the average of the PaymentTotal. Finally, ORDER BY is used to sort the result set by VendorName in ascending order.

Remark: DISTINCT keyword is used to not repeat a particular record again.

Q7) 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). *Use AP database.*

Ans: SELECT SUM(DerivedTable.MaxOfInvoices) AS LargestInvoicesUnpaid

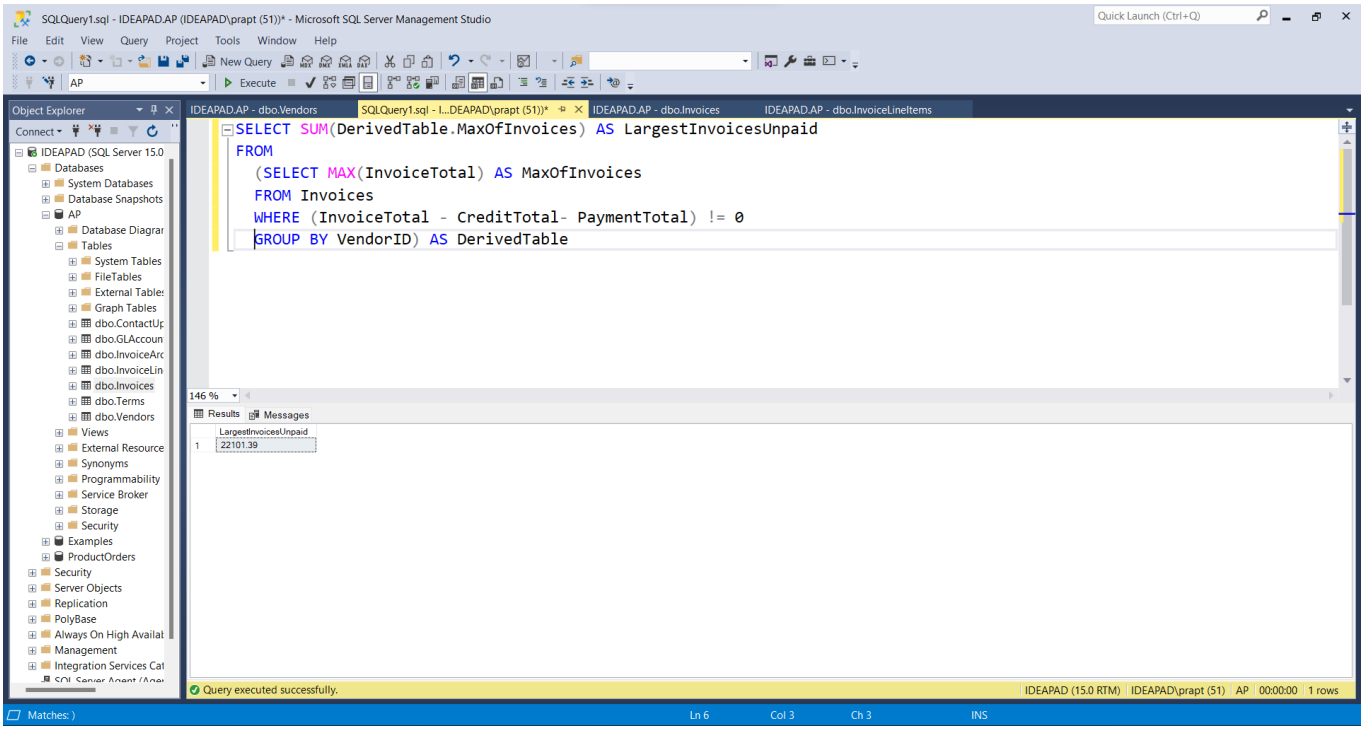
FROM

(SELECT MAX(InvoiceTotal) AS MaxOfInvoices

FROM Invoices

WHERE (InvoiceTotal - CreditTotal- PaymentTotal) != 0

GROUP BY VendorID) AS DerivedTable



Comment: Here, a derived table is made using a subquery inside the main query. The SELECT statement of the subquery is used to display the MaxOfInvoices column from the invoices table. This is obtained by using an aggregate function MAX() on the InvoiceTotal column. Such maximum values are to be displayed whose Balance is due I.e., the InvoiceTotal-CreditTotal-PayementTotal is not zero and thus, WHERE clause is used for applying this condition and the result set of this subqury is grouped by VendorID using the GROUP BY statement. The result set obtained using this subquery is stored in a derived table which is given an alias with AS keyword. SELECT statement is used to display the LargestInvoicesUnpaid column that is obtained by using aggregate function SUM() on the

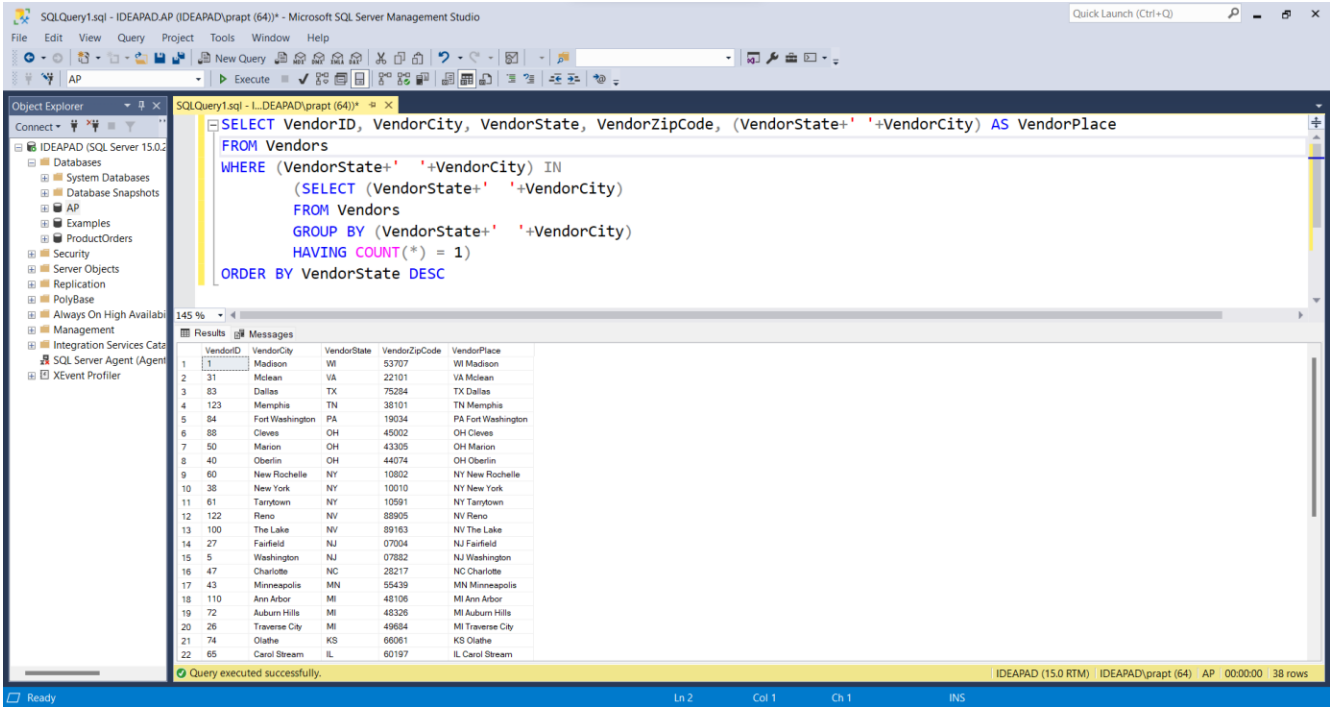
MaxOfInvoices column of the derived table. The subquery is put after the FROM of the main query.

Remark: Derived table is made by a subquery written after the FROM of the outer query. This table has its scope till the outer query is performed. This derived table is not created and saved in the database.

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

Ans: SELECT VendorID, VendorCity, VendorState, VendorZipCode, (VendorState+' '+VendorCity) AS VendorPlace

```
FROM Vendors
WHERE (VendorState+' '+VendorCity) IN
    (SELECT (VendorState+' '+VendorCity)
    FROM Vendors
    GROUP BY (VendorState+' '+VendorCity)
    HAVING COUNT(*) = 1)
ORDER BY VendorState DESC
```



Comment: Here, SELECT statement is used to display the VendorID, VendorCity, VendorState, VendorZipCode columns and the concatenation of VendorState and VendorCity from the Vendors table. WHERE clause is used to check the condition where the combination of unique state and unique city where no vendors with the same combination are displayed using the IN keyword that checks in the subquery. Subquery is used for the selection of the unique combination and grouping the records whose count is 1 I.e, not repeating and finally sort the result set by the VendorName column in descending order.

Remark: COUNT(*) is used to count the separate rows of the table. IN keyword is used to check whether a value is present in the list of the items given.

Remark for the lab: Concepts related to subqueries, summary queries are used including the basic keywords, clauses and statements in this lab.