

**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.**

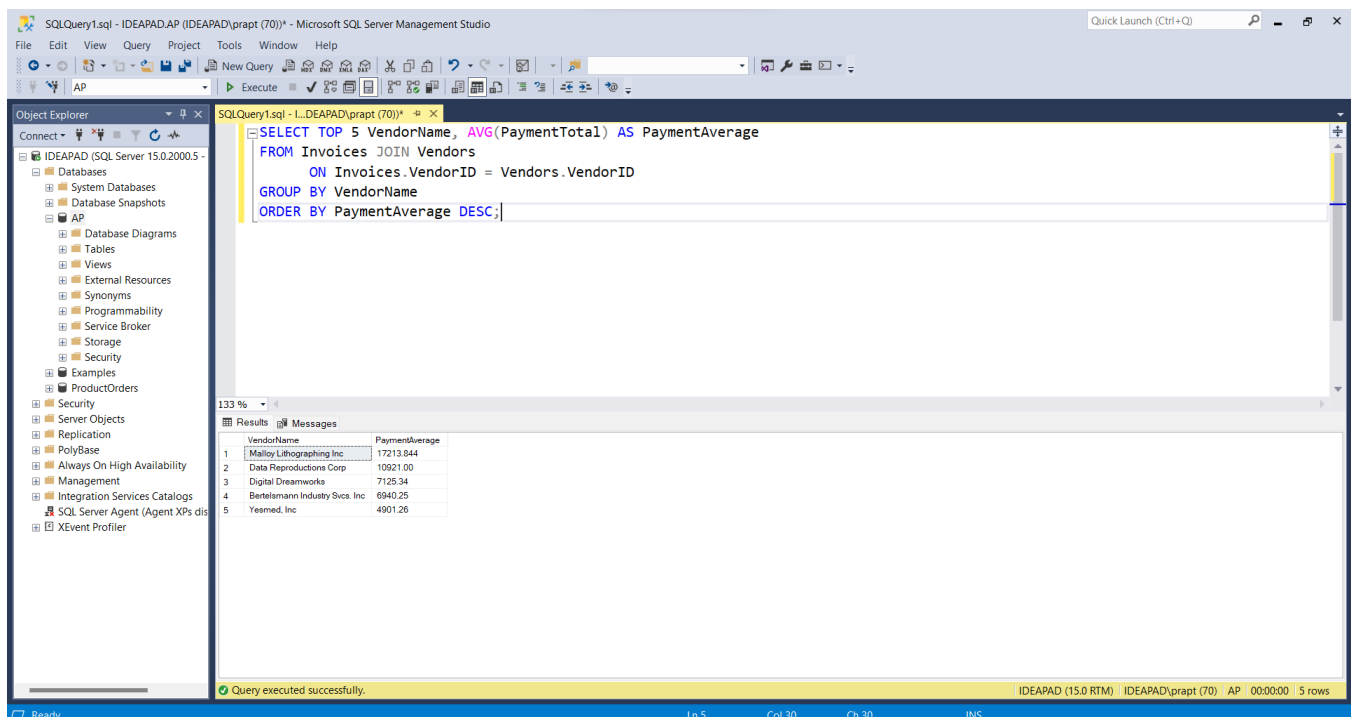
**Ans:** SELECT TOP 5 VendorName, AVG(PaymentTotal) AS PaymentAverage

FROM Invoices JOIN Vendors

ON Invoices.VendorID = Vendors.VendorID

GROUP BY VendorName

ORDER BY PaymentAverage DESC;



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

```
SELECT TOP 5 VendorName, AVG(PaymentTotal) AS PaymentAverage
FROM Invoices JOIN Vendors
ON Invoices.VendorID = Vendors.VendorID
GROUP BY VendorName
ORDER BY PaymentAverage DESC;
```

The Results pane displays the output of the query, showing the top 5 vendors by their average payment total:

	VendorName	PaymentAverage
1	Malloy Lithographing Inc.	17213.844
2	Data Reproductions Corp	10921.00
3	Digital Dreamworks	7125.34
4	Bertelsmann Industry Svcs. Inc.	6940.25
5	Yeemed, Inc.	4901.26

The status bar at the bottom indicates "Query executed successfully." and "5 rows".

**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, LinelItemCount, and LinelItemSum. LinelItemCount is the number of entries in the InvoiceLinelItems table that has that AccountNo. LinelItemSum is the sum of the InvoiceLinelItemAmount column for that AccountNo. Group the result set by account description, and sort it in ascending order of LinelItemSum. **Use AP database.**

**Ans:**     SELECT     AccountDescription,     COUNT(InvoiceLinelItems.AccountNo)     AS  
LinelItemCount,  
  
          SUM(InvoiceLinelItemAmount) AS LinelItemSum  
  
FROM GLAccounts JOIN InvoiceLinelItems  
  
          ON GLAccounts.AccountNo = InvoiceLinelItems.AccountNo  
  
GROUP BY AccountDescription  
  
ORDER BY LinelItemSum;

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

```
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;
```

The Results pane displays the output of the query, showing 21 rows of data. The columns are AccountDescription, LineItemCount, and LineItemSum.

AccountDescription	LineItemCount	LineItemSum
Utilities	1	16.82
Furniture	1	17.50
Meals	1	50.00
Office Supplies	3	175.80
Accounting	1	220.00
Telephone	7	266.01
Postage	1	290.00
Other Equipment	1	356.48
Building Maintenance	1	450.00
Travel and Accommodations	1	503.20
Group Insurance	3	564.00
Business Licenses and Taxes	1	856.92
UCI	1	1600.00
Building Lease	1	1750.00
Computer Equipment	3	2137.05
Direct Mail Advertising	6	3900.77
Books, Dues, and Subscriptions	6	5207.32
Book Production Costs	8	6175.12
Outside Services	3	13394.10
Freight	60	27599.65
Book Printing Costs	8	148759.97

The status bar at the bottom indicates the query was executed successfully.

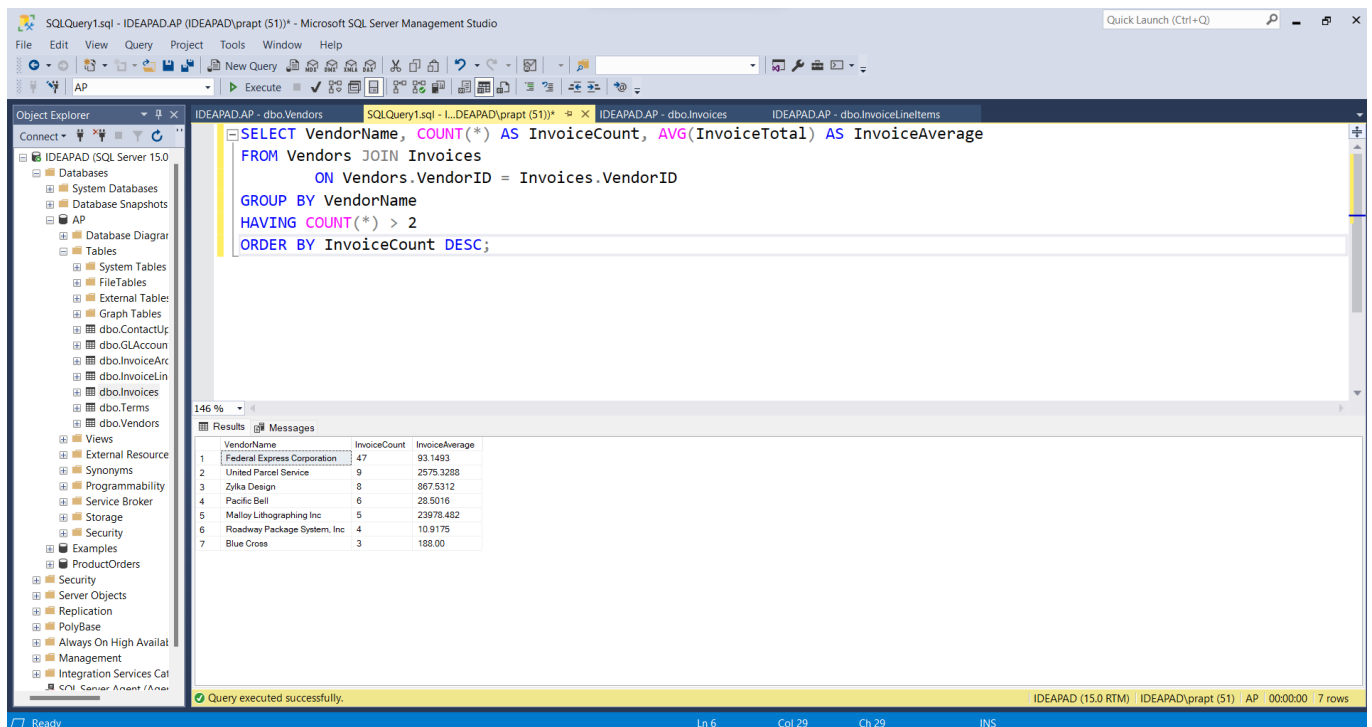
**Comment:** Here, SELECT statement is used to display the AccountDescription, LineItemCount and LineItemSum columns by joining the GLAccounts and InvoiceLineItems tables on AccountNo using JOIN and ON keywords. LineItemCount column is obtained by performing the aggregate COUNT() function on the AccountNo 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;
```

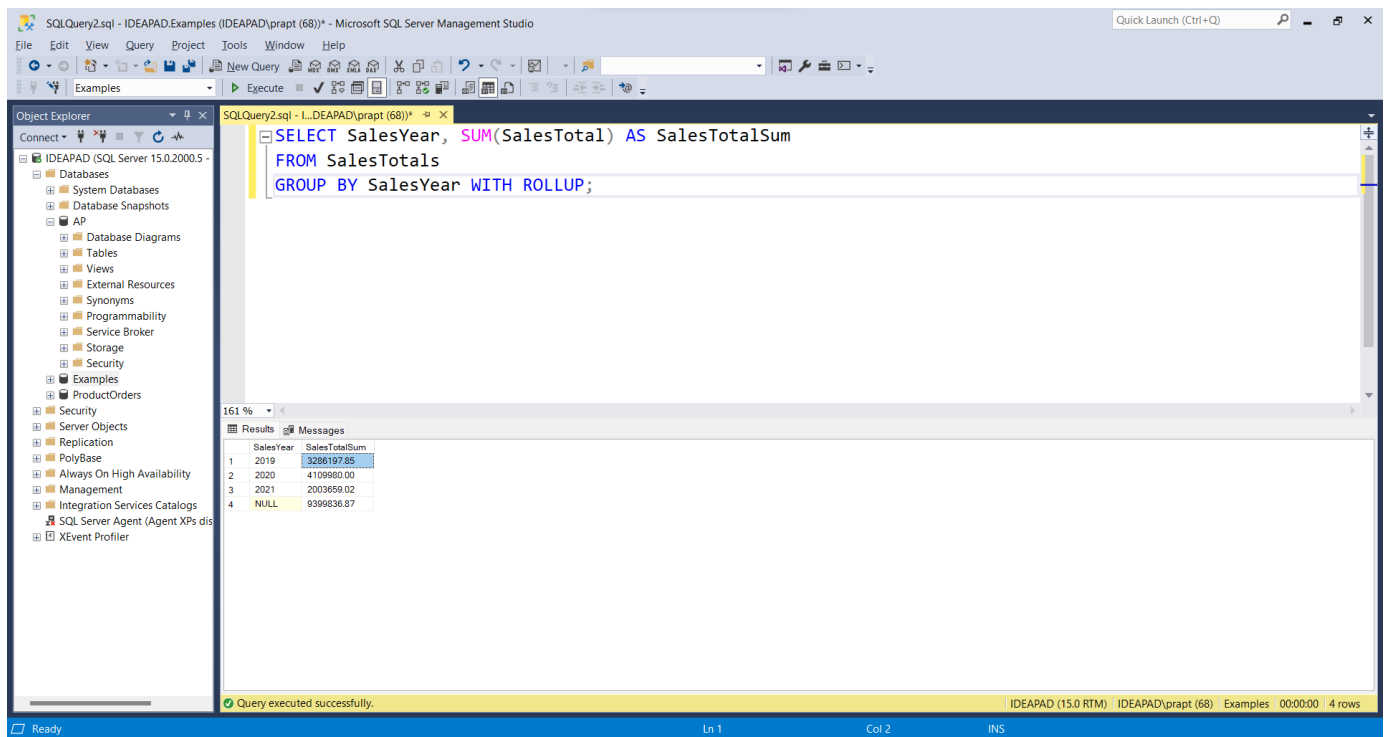


**Comment:** Here, SELECT statement is used to return VendorName, InvoiceCount and InvoiceAverage columns. InvoiceCount is obtained by applying the aggregate function COUNT(\*) to the table to get the number of invoices. InvoiceAverage column is obtained by applying the aggregate function AVG() to InvoiceTotal 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))

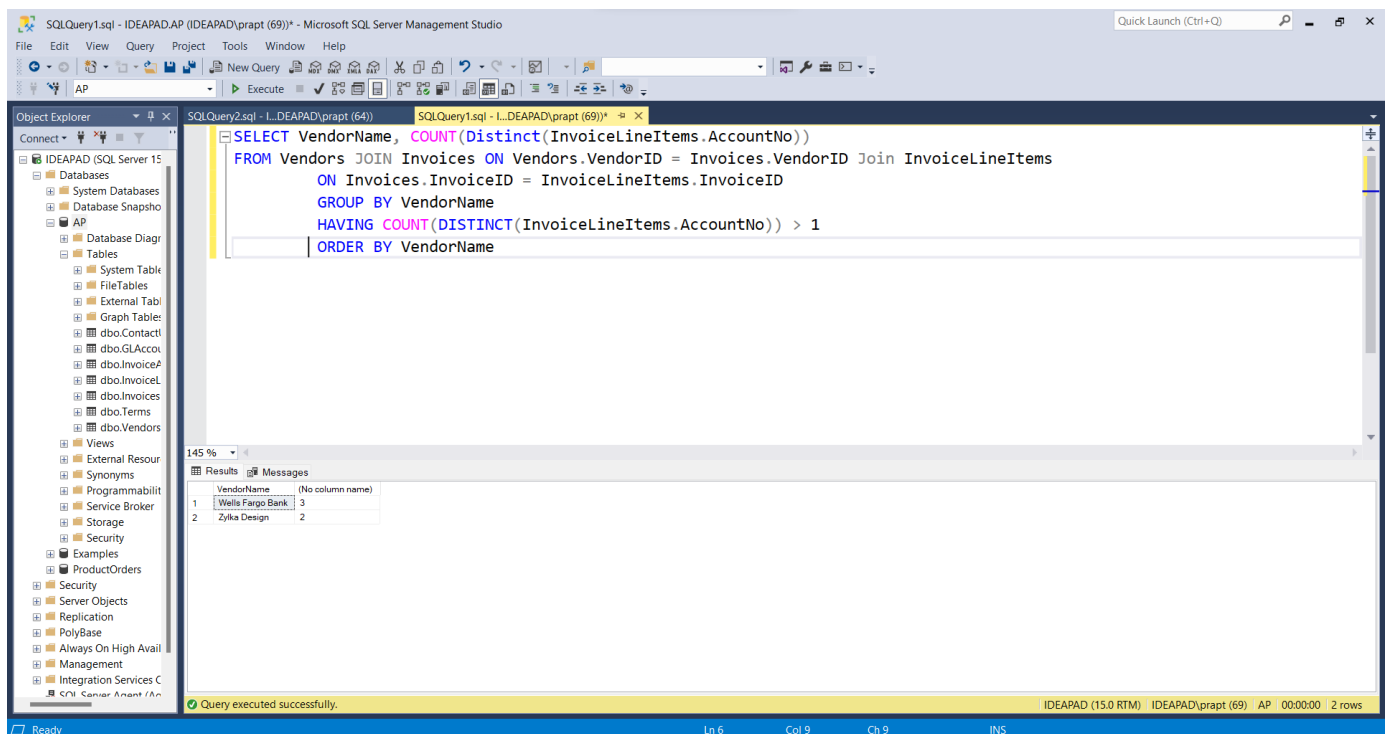
FROM Vendors JOIN Invoices ON Vendors.VendorID = Invoices.VendorID Join InvoiceLineItems

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)

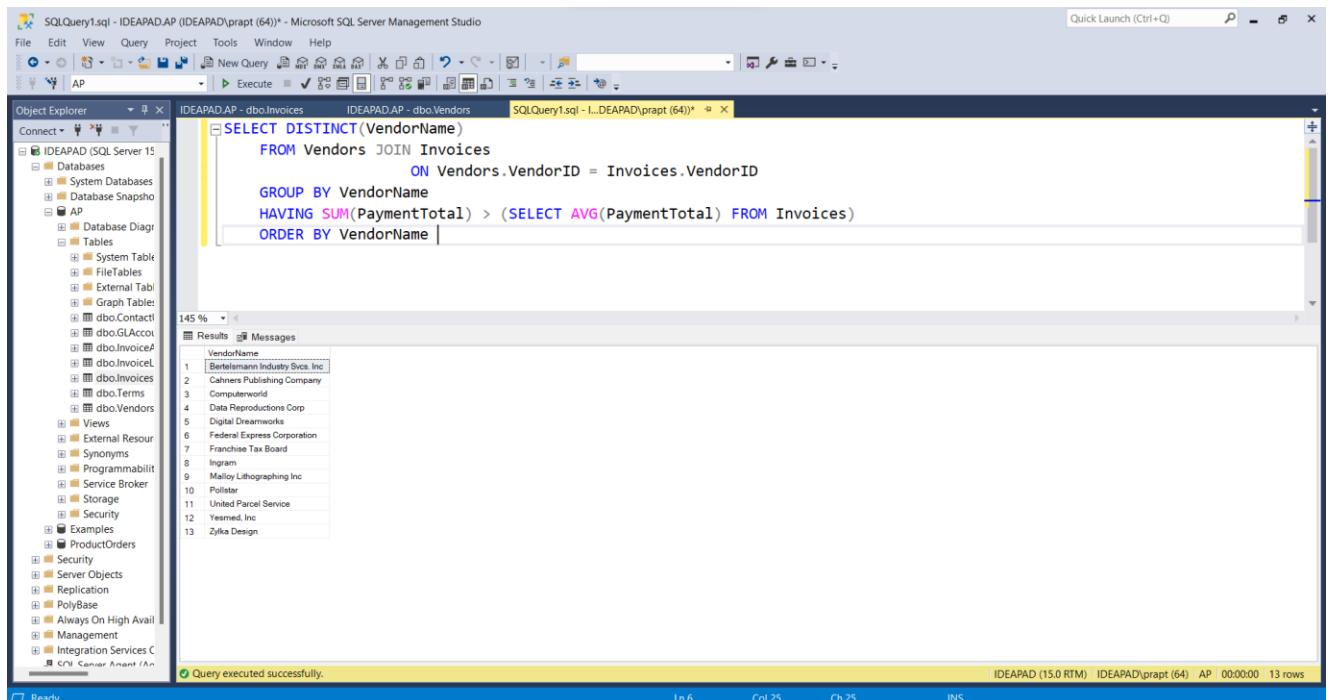
FROM Vendors JOIN Invoices

ON Vendors.VendorID = Invoices.VendorID

GROUP BY VendorName

HAVING SUM(PaymentTotal) > (SELECT AVG(PaymentTotal) FROM Invoices)

ORDER BY VendorName



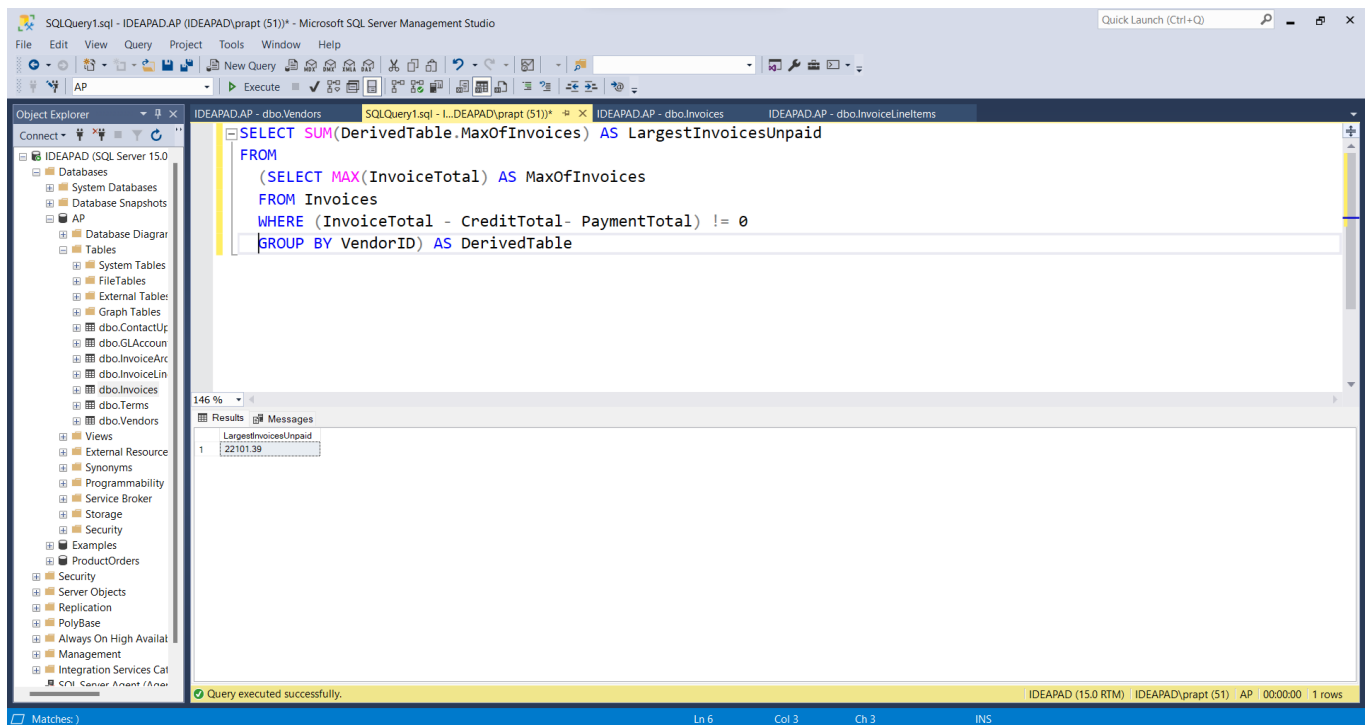
**Comment:** Here, SELECT statement is used to display the VendorName column with non-repeatable values for which DISTINCT keyword is used. JOIN keyword is used to join the two tables Vendors and Invoices on VendorID using ON keyword. Grouping is done using GROUP BY at VendorName column. A condition is applied to check whether the sum of 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-PaymentTotal is not zero and thus, WHERE clause is used for applying this condition and the result set of this subquery 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

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. The 'Object Explorer' on the left shows the database structure. The 'Query Editor' in the center contains the following SQL query:

```

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
  
```

The 'Results' pane at the bottom shows the output of the query, which is a table with 5 columns: VendorID, VendorCity, VendorState, VendorZipCode, and VendorPlace. The results are sorted by VendorState in descending order.

VendorID	VendorCity	VendorState	VendorZipCode	VendorPlace
1	Madison	WI	53707	WI Madison
31	McLean	VA	22101	VA McLean
83	Dallas	TX	75284	TX Dallas
123	Memphis	TN	38101	TN Memphis
84	Fort Washington	PA	19034	PA Fort Washington
88	Cleveland	OH	45002	OH Cleveland
50	Marion	OH	43305	OH Marion
40	Oberlin	OH	44074	OH Oberlin
60	New Rochelle	NY	10802	NY New Rochelle
38	New York	NY	10010	NY New York
61	Tarrytown	NY	10591	NY Tarrytown
122	Reno	NV	89505	NV Reno
100	The Lake	NV	89163	NV The Lake
27	Fairfield	NJ	07004	NJ Fairfield
5	Washington	NJ	07882	NJ Washington
47	Charlotte	NC	28217	NC Charlotte
43	Minneapolis	MN	55439	MN Minneapolis
110	Ann Arbor	MI	48106	MI Ann Arbor
72	Auburn Hills	MI	48326	MI Auburn Hills
26	Traverse City	MI	49684	MI Traverse City
74	Olathe	KS	66061	KS Olathe
65	Carol Stream	IL	60197	IL Carol Stream

The status bar at the bottom indicates 'Query executed successfully.' and shows the execution time as 00:00:00 with 38 rows returned.

**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.