

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.

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the Object Explorer with the 'ProductOrders' database selected. The right pane shows a SQL query window with the following query:

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

Below the query window, the 'Results' tab displays the following data:

Artist	TotalEarnings
1 Jess & Odie	13.00
2 The Ubemerds	13.00
3 Sewed the Vest Pocket	16.95
4 Onn & Onn	17.00
5 No Rest For The Weary	34.90

The status bar at the bottom indicates 'Query executed successfully.' and 'DESKTOP-NUN5QT7 (15.0 RTM) | DESKTOP-NUN5QT7 (15.0 RTM) | ProductOrders | 100.00 | 5 rows'.

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

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

```
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---
```

The Results pane displays the following data:

	VendorName	InvoiceCount	InvoiceAverage
1	Federal Express Corporation	47	93.1493
2	United Parcel Service	9	2575.3288
3	Zylka Design	8	867.5312
4	Pacific Bell	6	28.5016
5	Malloy Lithographing Inc	5	23978.482
6	Roadway Package System, Inc	4	10.9175
7	Blue Cross	3	188.00
8	Cardinal Business Media, Inc.	2	132.68
9	Compuserve	2	9.95
10	Data Reproductions Corp	2	10963.655
11	IBM	2	600.06

The status bar at the bottom indicates: "Query executed successfully. DESKTOP-NUN5QT7 (15.0 RTM) DESKTOP-NUN5QT7\rhpri... AP 00:00:00 34 rows".

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 LineItemSum 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"

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

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

The Results pane displays the following data:

	AccountDescription	LineItemCount	LineItemSum
1	Book Printing Costs	8	148759.97
2	Freight	60	27599.65
3	Outside Services	3	13394.10
4	Book Production Costs	8	6175.12
5	Books, Dues, and Subscriptions	6	5207.32
6	Direct Mail Advertising	6	3900.77
7	Computer Equipment	3	2137.05
8	Group Insurance	3	564.00
9	Telephone	7	266.01
10	Office Supplies	3	175.80

The status bar at the bottom indicates: Query executed successfully. DESKTOP-NUN5QT7 (15.0 RTM) DESKTOP-NUN5QT7\rhpri... AP 00:00:00 10 rows

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.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure, including tables like `dbo.InvoiceLineItems` and `dbo.Invoices`. The main query window shows the following SQL query:

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

The Results pane at the bottom displays the output of the query, showing a list of accounts and their corresponding total invoice amounts. The data is as follows:

	AccountNo	LineitemSum
1	150	17.50
2	160	2137.05
3	170	356.48
4	400	148759.97
5	403	6175.12
6	507	1600.00
7	510	564.00
8	520	1750.00
9	521	16.62
10	522	266.01
11	523	450.00

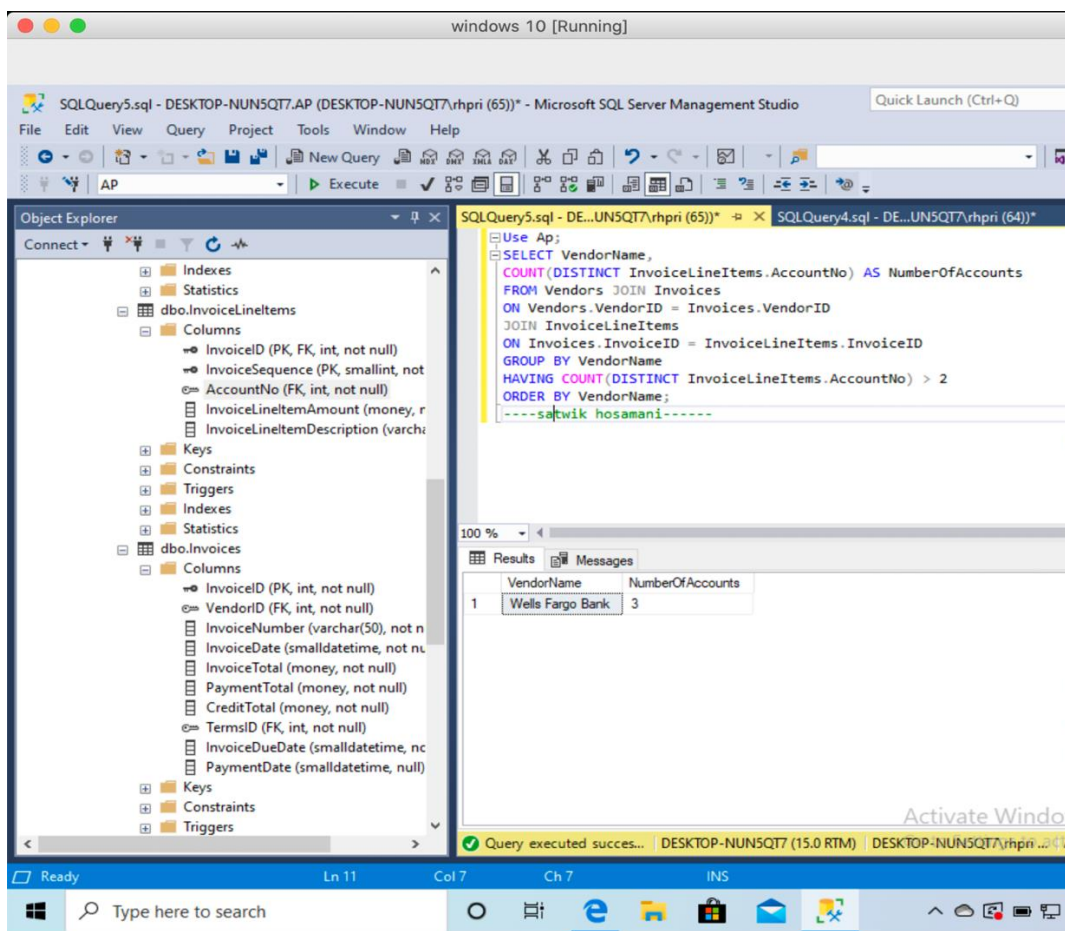
The status bar at the bottom indicates that the query was executed successfully, returning 22 rows.

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 tables 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 invoiceitems (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 invoiceitems. 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 invoiceitems. AVG function is used for calculating the average.

The screenshot displays the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including tables like `dbo.InvoiceLineItems` and `dbo.Invoices`. The main query window shows the following SQL query:

```
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;
-----satwik hosamani-----
```

The Results pane at the bottom shows the output of the query, which is a list of distinct AccountNo values:

AccountNo
400
540
553
572
589

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

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.

The screenshot displays the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including tables like `dbo.InvoiceLineItems` and `dbo.Invoices`. The main query window shows the following SQL code:

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

The Results pane at the bottom shows the output of the query:

SumOfMINimums
12489.10

The status bar at the bottom indicates that the query was executed successfully, returning 1 row.

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 in the main select statement VendorCity and VendorZipCode are filtered using the not in function. COUNT function is used to calculate the number.

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor displays the following SQL code:

```
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-----
```

The Results pane shows the output of the query, which is a table with 4 columns: VendorID, VendorCity, VendorState, and VendorZipCode. The results are sorted by VendorCity in descending order and VendorState in ascending order.

VendorID	VendorCity	VendorState	VendorZipCode
82	Washington	DC	20006
2	Washington	DC	20090
3	Washington	DC	20559
24	Washington	IA	52353
5	Washington	NJ	07882
99	Valencia	CA	91355
89	Turlock	CA	95380
26	Traverse City	MI	49684
100	The Lake	NV	89163
61	Tarrytown	NY	10591
21	St Louis	MO	63105

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.