



#Query 1. Write a query that shows which customer countries have cancelled orders. List the #country names, the number of orders that country has cancelled, and the number of unique products in #those cancelled orders. Order the results by the number of products descending.

```

use ClassicModels;
select distinct country, count(distinct OrderDetails.orderNumber) as numCancelled,
count(distinct OrderDetails.productCode) as numUnique from Customers
join Orders on Customers.customerNumber = Orders.customerNumber
join OrderDetails on OrderDetails.orderNumber = Orders.orderNumber
where status = 'Cancelled'
group by country
order by numUnique;
  
```

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'SCHEMAS' tree with various database objects. The central query window contains two queries. Query 1 is a complex join query that filters for cancelled orders and counts distinct countries, cancelled orders, and unique products. Query 2 is a simple query to find customers who have not made any payments. The Results pane at the bottom shows the output of Query 1 as a table with columns: country, numCancelled, numUnique. The table contains 5 rows of data for different countries.

country	numCancelled	numUnique
UK	1	14
USA	1	14
Spain	1	16
Sweden	1	16
New Zealand	2	19

The Action Output pane at the bottom shows the execution of three queries, including the one for customers with no payments.

#Query 2. Write a query to display the customer name for all customers that have not made any #payments. Order the results by customer name in the ascending order.

```
select customerName from Customers
where not exists (select * from Payments where Payments.customerNumber =
Customers.customerNumber)
order by customerName asc;
```

The screenshot shows a MySQL IDE interface. The left sidebar displays a 'SCHEMAS' tree with various database objects like Customers, Employees, Offices, etc. The main query editor contains the following SQL code:

```

9  join Orders on Customers.customerNumber = Orders.customerNumber
10 join OrderDetails on OrderDetails.orderNumber = Orders.orderNumber
11 where status = 'Cancelled'
12 group by country
13 order by numUnique;
14
15 #Query 2. Write a query to display the customer name for all customers that have not made any
16 #payments. Order the results by customer name in the ascending order.
17
18 select customerName from Customers
19 where not exists (select * from Payments where Payments.customerNumber = Customers.customerNumber)
20 order by customerName asc;
21
22
23 #Query 3. Write a query to list the different offices (their address), the count of the number of
24 #employees at that office (who have helped make an order) and the number of orders that
25 #employees at that office were sales representatives for.

```

Below the query editor, the 'Result Grid' shows the results of the executed query (Query 25). The grid has two columns: 'customerName' and an empty column. The results are as follows:

customerName	
American Souvenirs Inc	
ANG Resellers	
Antion Designs Ltd	
Asian Shopping Network, Co	
Asian Treasures, Inc.	
BG&E Collectables	
Cramer Spezialit	
Dor Hund Imports	
Foster Online Stores, Inc	
Franken Gifts, Co	

The 'Action Output' section at the bottom shows the execution log:

Time	Action	Response
253 19:35:54	select Offices.addressLine1, count(distinct employeeNumber), count(distinct orderNumber) from Offices join Employee...	7 row(s) returned
254 19:35:54	use ClassicModels	0 row(s) affected
255 19:35:54	select firstName, lastName, concat('\$', round(sum(amount), 2)) as revenue from Employees join Customers on Custo...	15 row(s) returned

#Query 3. Write a query to list the different offices (their address), the count of the number of #employees at that office (who have helped make an order) and the number of orders that #employees at that office were sales representatives for.

```

select Offices.addressLine1, count(distinct employeeNumber), count(distinct orderNumber) from
Offices
join Employees on Employees.officeCode = Offices.officeCode
join Customers on Customers.salesRepEmployeeNumber = Employees.employeeNumber
join Orders on Orders.customerNumber = Customers.customerNumber
where Customers.salesRepEmployeeNumber = Employees.employeeNumber
group by Offices.addressLine1;

```

The screenshot shows a MySQL IDE interface. The top toolbar includes icons for home, back, forward, and other navigation functions. The main window is divided into three panes: a left sidebar for 'SCHEMAS' (showing a tree view of databases like Customers, Employees, etc.), a central 'Query Editor' with SQL code, and a bottom pane for 'Result Grid' and 'Action Output'.

Query Editor Content:

```

19 where not exists (select * from Payments where Payments.customerNumber = Customers.customerNumber)
20 order by customerName asc;
21
22
23
24 #Query 3. Write a query to list the different offices (their address), the count of the number of
25 #employees at that office (who have helped make an order) and the number of orders that
26 #employees at that office were sales representatives for.
27
28 select Offices.addressLine1, count(distinct employeeNumber), count(distinct orderNumber) from Offices
29 join Employees on Employees.officeCode = Offices.officeCode
30 join Customers on Customers.salesRepEmployeeNumber = Employees.employeeNumber
31 join Orders on Orders.customerNumber = Customers.customerNumber
32 where Customers.salesRepEmployeeNumber = Employees.employeeNumber
33 group by Offices.addressLine1;
34
35

```

Result Grid:

addressLine1	count(distinct employeeNum...	count(distinct orderNum...
100 Market Street	2	48
1550 Court Place	2	32
25 Old Broad Street	2	47
4-1 Kiolicho	1	16
43 Rue Joffroy D'abbans	4	106
5-11 Wentworth Avenue	2	38
523 East 53rd Street	2	39

Action Output:

Time	Action	Response
253 19:35:54	select Offices.addressLine1, count(distinct employeeNumber), count(distinct orderNumber) from Offices join Employees on Employees.officeCode = Offices.officeCode	7 row(s) returned
254 19:35:54	use ClassicModels	0 row(s) affected
255 19:35:54	select firstName, lastName, concat('\$', round(sum(amount), 2)) as revenue from Employees join Customers on Customers.salesRepEmployeeNumber = Employees.employeeNumber	15 row(s) returned

#Query 4. Write a query to display the name (last name and first name) of the employee and the #amount of revenue he/she has generated (i.e. how many dollars (Payments) has been received by #their customers). Give this column a new name. Round the value to 2 decimals, concatenate the #dollar symbol on the column, and order the results in descending value of the revenue received.

use ClassicModels;
select firstName, lastName, concat('\$', round(sum(amount), 2)) as revenue from Employees
join Customers on Customers.salesRepEmployeeNumber = Employees.employeeNumber
join Payments on Payments.customerNumber = Customers.customerNumber
group by Employees.employeeNumber
order by sum(amount) desc;

Terry College ServerMySQL ModelEER Diagram1

AdministrationSchemasassignment 3a3q1a3q2a3q3*a3q3-2a3q4feb 17 stuffexists: not exists comand

SCHEMASFilter objects

CustomersEmployeesOfficesOrderDetailsOrdersColumnsIndexesForeign KeysTriggersPaymentsProductLinesProductsViewsStored ProceduresFunctionsGeographyTextTablesalienassemblycardeptdonorempexpedgiftitemlineitemmonarchnationperson

```
33 group by Offices.addressLine1;
34
35
36 #Query 4. Write a query to display the name (last name and first name) of the employee and the
37 #amount of revenue he/she has generated (i.e. how many dollars (Payments) has been received by
38 #their customers). Give this column a new name. Round the value to 2 decimals, concatenate the
39 #dollar symbol on the column, and order the results in descending value of the revenue received.
40
41
42 use ClassicModels;
43 select firstName, lastName, concat('$', round(sum(amount), 2)) as revenue from Employees
44 join Customers on Customers.salesRepEmployeeNumber = Employees.employeeNumber
45 join Payments on Payments.customerNumber = Customers.customerNumber
46 group by Employees.employeeNumber
47 order by sum(amount) desc;
48
49
```

100%20:13

Result GridFilter Rows: SearchExport:

firstName	lastName	revenue
Gerard	Hernandez	\$1112003.81
Leslie	Jennings	\$689506.55
Pamela	Castillo	\$150201.87
Larry	Bott	\$686653.25
Barry	Jones	\$637672.65
George	Vanaul	\$584406.8
Loui	Bondur	\$569485.75
Andy	Fyter	\$509385.82
Peter	Marsh	\$497907.16
Foon Yue	Tsang	\$488212.67

Result 24Customers 25Result 26Result 27Read Only

Action Output

	Time	Action	Response
✓	253 19:35:54	select Offices.addressLine1, count(distinct employeeNumber), count(distinct orderNumber) from Offices join Employee...	7 row(s) returned
✓	254 19:35:54	use ClassicModels	0 row(s) affected
✓	255 19:35:54	select firstName, lastName, concat('\$', round(sum(amount), 2)) as revenue from Employees join Customers on Custo...	15 row(s) returned

Query Completed