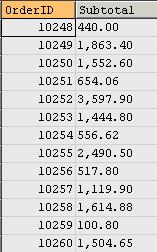
**Install northwind database**

**1. Order Subtotals**

For each order, calculate a subtotal for each Order (identified by OrderID). This is a simple query using GROUP BY to aggregate data for each order.

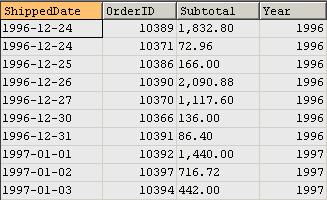
-- Get subtotal for each order.  
select OrderID,   
 format(sum(UnitPrice \* Quantity \* (1 - Discount)), 2) as Subtotal  
from order\_details  
group by OrderID  
order by OrderID;

Here is the query result. 830 records returned.  


**2. Sales by Year**

This query shows how to get the year part from Shipped\_Date column. A subtotal is calculated by a sub-query for each order. The sub-query forms a table and then joined with the Orders table.

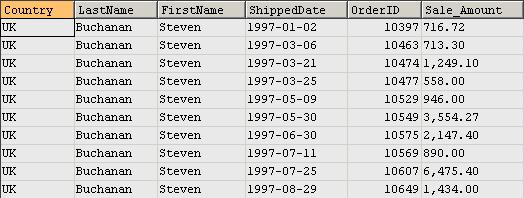
select distinct date(a.ShippedDate) as ShippedDate,   
 a.OrderID,   
 b.Subtotal,   
 year(a.ShippedDate) as Year  
from Orders a   
inner join  
(  
 -- Get subtotal for each order  
 select distinct OrderID,   
 format(sum(UnitPrice \* Quantity \* (1 - Discount)), 2) as Subtotal  
 from order\_details  
 group by OrderID   
) b on a.OrderID = b.OrderID  
where a.ShippedDate is not null  
 and a.ShippedDate between date('1996-12-24') and date('1997-09-30')  
order by a.ShippedDate;

Here is the query result. 296 records returned.  


**3. Employee Sales by Country**

For each employee, get their sales amount, broken down by country name.

select distinct b.\*, a.CategoryName  
from Categories a   
inner join Products b on a.CategoryID = b.CategoryID  
where b.Discontinued = 'N'  
order by b.ProductName;

Here is the query result. 296 records returned.  


**4. Alphabetical List of Products**

This is a rather simple query to get an alphabetical list of products.

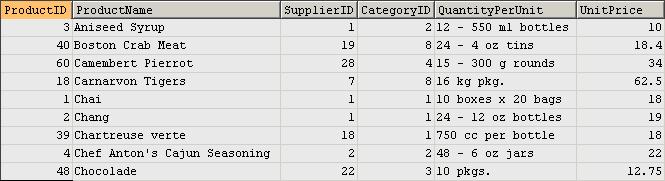
select distinct b.\*, a.Category\_Name

from Categories a

inner join Products b on a.Category\_ID = b.Category\_ID

where b.Discontinued = 'N'

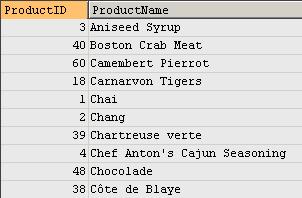
order by b.Product\_Name;

Here is the query result. 69 records returned.  


**5. Current Product List**

This is another simple query. No aggregation is used for summarizing data.

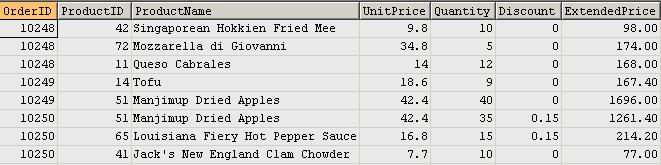
select ProductID, ProductName  
from products  
where Discontinued = 'N'  
order by ProductName;

Here is the query result. 69 records returned.  


**6. Order Details Extended**

This query calculates sales price for each order after discount is applied.

select distinct y.OrderID,   
 y.ProductID,   
 x.ProductName,   
 y.UnitPrice,   
 y.Quantity,   
 y.Discount,   
 round(y.UnitPrice \* y.Quantity \* (1 - y.Discount), 2) as ExtendedPrice  
from Products x  
inner join Order\_Details y on x.ProductID = y.ProductID  
order by y.OrderID;

Here is the query result. 2,155 records returned.  


**7. Sales by Category**

For each category, we get the list of products sold and the total sales amount. Note that, in the second query, the inner query for table c is to get sales for each product on each order. It then joins with outer query on Product\_ID. In the outer query, products are grouped for each category.

/\*  
Query 1: normal joins  
\*/  
select distinct a.CategoryID,   
 a.CategoryName,   
 b.ProductName,   
 sum(round(y.UnitPrice \* y.Quantity \* (1 - y.Discount), 2)) as ProductSales  
from Order\_Details y  
inner join Orders d on d.OrderID = y.OrderID  
inner join Products b on b.ProductID = y.ProductID  
inner join Categories a on a.CategoryID = b.CategoryID  
where d.OrderDate between date('1997/1/1') and date('1997/12/31')  
group by a.CategoryID, a.CategoryName, b.ProductName  
order by a.CategoryName, b.ProductName, ProductSales;  
   
/\*  
Query 2: join with a sub query  
   
This query returns identical result as above, but here  
sub query is used to calculate extended price which   
then used in the main query to get ProductSales  
\*/   
select distinct a.CategoryID,   
 a.CategoryName,   
 b.ProductName,   
 sum(c.ExtendedPrice) as ProductSales  
from Categories a   
inner join Products b on a.CategoryID = b.CategoryID  
inner join   
(  
 select distinct y.OrderID,   
 y.ProductID,   
 x.ProductName,   
 y.UnitPrice,   
 y.Quantity,   
 y.Discount,   
 round(y.UnitPrice \* y.Quantity \* (1 - y.Discount), 2) as ExtendedPrice  
 from Products x  
 inner join Order\_Details y on x.ProductID = y.ProductID  
 order by y.OrderID  
) c on c.ProductID = b.ProductID  
inner join Orders d on d.OrderID = c.OrderID  
where d.OrderDate between date('1997/1/1') and date('1997/12/31')  
group by a.CategoryID, a.CategoryName, b.ProductName  
order by a.CategoryName, b.ProductName, ProductSales;

Here is the query result. 77 records returned.  


**8. Ten Most Expensive Products**

The two queries below return the same result. It demonstrates how MySQL limits the number of records returned.

The first query uses correlated sub-query to get the top 10 most expensive products.

The second query retrieves data from an ordered sub-query table and then the keyword LIMIT is used outside the sub-query to restrict the number of rows returned.

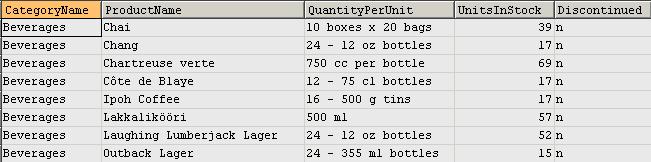
-- Query 1  
select distinct ProductName as Ten\_Most\_Expensive\_Products,   
 UnitPrice  
from Products as a  
where 10 >= (select count(distinct UnitPrice)  
 from Products as b  
 where b.UnitPrice >= a.UnitPrice)  
order by UnitPrice desc;  
   
-- Query 2  
select \* from  
(  
 select distinct ProductName as Ten\_Most\_Expensive\_Products,   
 UnitPrice  
 from Products  
 order by UnitPrice desc  
) as a  
limit 10;

Here is the query result. 10 records returned.  


**9. Products by Category**

This is a simple query just because it's in Access Northwind so we converted it here in MySQL.

select distinct a.CategoryName,   
 b.ProductName,   
 b.QuantityPerUnit,   
 b.UnitsInStock,   
 b.Discontinued  
from Categories a  
inner join Products b on a.CategoryID = b.CategoryID  
where b.Discontinued = 'N'  
order by a.CategoryName, b.ProductName;

Here is the query result. 69 records returned.  


**10. Customers and Suppliers by City**

This query shows how to use UNION to merge Customers and Suppliers into one result set by identifying them as having different relationships to Northwind Traders - Customers and Suppliers.

select City, CompanyName, ContactName, 'Customers' as Relationship   
from Customers  
union  
select City, CompanyName, ContactName, 'Suppliers'  
from Suppliers  
order by City, CompanyName;

Here is the query result. 120 records returned.  


**11. Products Above Average Price**

This query shows how to use sub-query to get a single value (average unit price) that can be used in the outer-query.

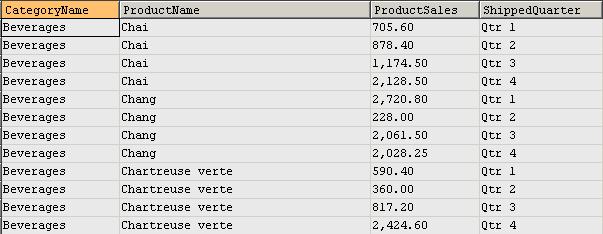
select distinct ProductName, UnitPrice  
from Products  
where UnitPrice > (select avg(UnitPrice) from Products)  
order by UnitPrice;

Here is the query result. 25 records returned.  


**12. Product Sales for 1997**

This query shows how to group categories and products by quarters and shows sales amount for each quarter.

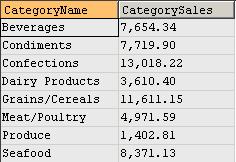
select distinct a.CategoryName,   
 b.ProductName,   
 format(sum(c.UnitPrice \* c.Quantity \* (1 - c.Discount)), 2) as ProductSales,  
 concat('Qtr ', quarter(d.ShippedDate)) as ShippedQuarter  
from Categories a  
inner join Products b on a.CategoryID = b.CategoryID  
inner join Order\_Details c on b.ProductID = c.ProductID  
inner join Orders d on d.OrderID = c.OrderID  
where d.ShippedDate between date('1997-01-01') and date('1997-12-31')  
group by a.CategoryName,   
 b.ProductName,   
 concat('Qtr ', quarter(d.ShippedDate))  
order by a.CategoryName,   
 b.ProductName,   
 ShippedQuarter;

Here is the query result. 286 records returned.  


**13. Category Sales for 1997**

This query shows sales figures by categories - mainly just aggregation with sub-query. The inner query aggregates to product level, and the outer query further aggregates the result set from inner-query to category level.

select CategoryName, format(sum(ProductSales), 2) as CategorySales  
from  
(  
 select distinct a.CategoryName,   
 b.ProductName,   
 format(sum(c.UnitPrice \* c.Quantity \* (1 - c.Discount)), 2) as ProductSales,  
 concat('Qtr ', quarter(d.ShippedDate)) as ShippedQuarter  
 from Categories as a  
 inner join Products as b on a.CategoryID = b.CategoryID  
 inner join Order\_Details as c on b.ProductID = c.ProductID  
 inner join Orders as d on d.OrderID = c.OrderID   
 where d.ShippedDate between date('1997-01-01') and date('1997-12-31')  
 group by a.CategoryName,   
 b.ProductName,   
 concat('Qtr ', quarter(d.ShippedDate))  
 order by a.CategoryName,   
 b.ProductName,   
 ShippedQuarter  
) as x  
group by CategoryName  
order by CategoryName;

Here is the query result. 8 records returned.  


**14. Quarterly Orders by Product**

This query shows how to convert order dates to the corresponding quarters. It also demonstrates how SUM function is used together with CASE statement to get sales for each quarter, where quarters are converted from OrderDate column.

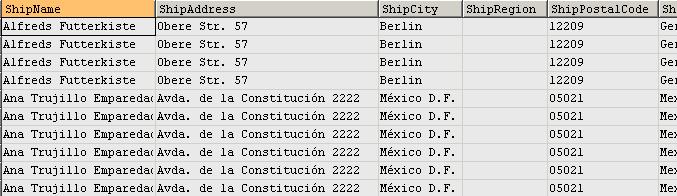
select a.ProductName,   
 d.CompanyName,   
 year(OrderDate) as OrderYear,  
 format(sum(case quarter(c.OrderDate) when '1'   
 then b.UnitPrice\*b.Quantity\*(1-b.Discount) else 0 end), 0) "Qtr 1",  
 format(sum(case quarter(c.OrderDate) when '2'   
 then b.UnitPrice\*b.Quantity\*(1-b.Discount) else 0 end), 0) "Qtr 2",  
 format(sum(case quarter(c.OrderDate) when '3'   
 then b.UnitPrice\*b.Quantity\*(1-b.Discount) else 0 end), 0) "Qtr 3",  
 format(sum(case quarter(c.OrderDate) when '4'   
 then b.UnitPrice\*b.Quantity\*(1-b.Discount) else 0 end), 0) "Qtr 4"   
from Products a   
inner join Order\_Details b on a.ProductID = b.ProductID  
inner join Orders c on c.OrderID = b.OrderID  
inner join Customers d on d.CustomerID = c.CustomerID   
where c.OrderDate between date('1997-01-01') and date('1997-12-31')  
group by a.ProductName,   
 d.CompanyName,   
 year(OrderDate)  
order by a.ProductName, d.CompanyName;

Here is the query result. 947 records returned.  


**15. Invoice**

A simple query to get detailed information for each sale so that invoice can be issued.

select distinct b.ShipName,   
 b.ShipAddress,   
 b.ShipCity,   
 b.ShipRegion,   
 b.ShipPostalCode,   
 b.ShipCountry,   
 b.CustomerID,   
 c.CompanyName,   
 c.Address,   
 c.City,   
 c.Region,   
 c.PostalCode,   
 c.Country,   
 concat(d.FirstName, ' ', d.LastName) as Salesperson,   
 b.OrderID,   
 b.OrderDate,   
 b.RequiredDate,   
 b.ShippedDate,   
 a.CompanyName,   
 e.ProductID,   
 f.ProductName,   
 e.UnitPrice,   
 e.Quantity,   
 e.Discount,  
 e.UnitPrice \* e.Quantity \* (1 - e.Discount) as ExtendedPrice,  
 b.Freight  
from Shippers a   
inner join Orders b on a.ShipperID = b.ShipVia   
inner join Customers c on c.CustomerID = b.CustomerID  
inner join Employees d on d.EmployeeID = b.EmployeeID  
inner join Order\_Details e on b.OrderID = e.OrderID  
inner join Products f on f.ProductID = e.ProductID  
order by b.ShipName;

Here is the query result. 2,155 records returned.  


**16. Number of units in stock by category and supplier continent**

This query shows that case statement is used in GROUP BY clause to list the number of units in stock for each product category and supplier's continent. Note that, if only s.Country (not the case statement) is used in the GROUP BY, duplicated rows will exist for each product category and supplier continent.

select c.CategoryName as "Product Category",   
 case when s.Country in   
 ('UK','Spain','Sweden','Germany','Norway',  
 'Denmark','Netherlands','Finland','Italy','France')  
 then 'Europe'  
 when s.Country in ('USA','Canada','Brazil')   
 then 'America'  
 else 'Asia-Pacific'  
 end as "Supplier Continent",   
 sum(p.UnitsInStock) as UnitsInStock  
from Suppliers s   
inner join Products p on p.SupplierID=s.SupplierID  
inner join Categories c on c.CategoryID=p.CategoryID   
group by c.CategoryName,   
 case when s.Country in   
 ('UK','Spain','Sweden','Germany','Norway',  
 'Denmark','Netherlands','Finland','Italy','France')  
 then 'Europe'  
 when s.Country in ('USA','Canada','Brazil')   
 then 'America'  
 else 'Asia-Pacific'  
 end;

Here is the query result. 21 records returned.  
