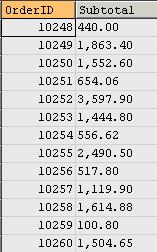
**Install northwind database**

**1. Order Subtotals**

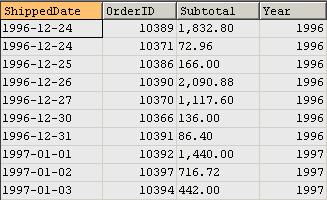
For each order, calculate a subtotal for each Order (identified by OrderID). This is a simple query using GROUP BY to aggregate data for each order.

-- Get subtotal for each order.

Here is the query result. 830 records returned.  


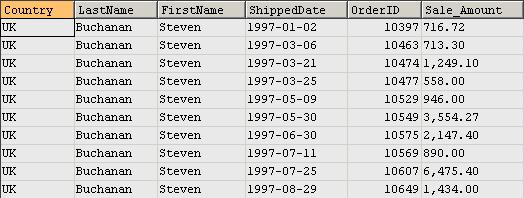
**2. Sales by Year**

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.

Here is the query result. 296 records returned.  


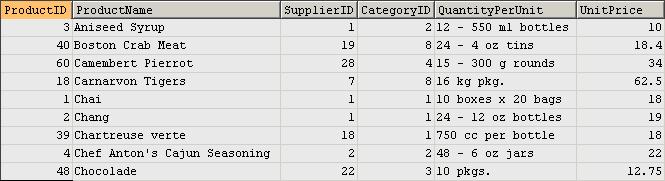
**3. Employee Sales by Country**

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

Here is the query result. 296 records returned.  


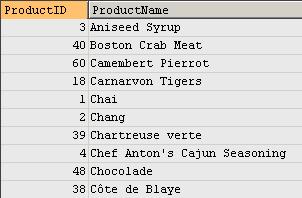
**4. Alphabetical List of Products**

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

Here is the query result. 69 records returned.  


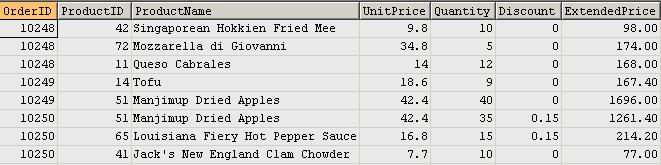
**5. Current Product List**

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

Here is the query result. 69 records returned.  


**6. Order Details Extended**

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

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  
\*/  
  
/\*  
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  
\*/

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.

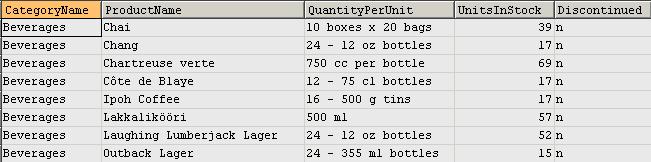
The second query retrieves data from an ordered sub-query table and then the keyword LIMIT is used outside the sub-query to restrict the number of rows returned.

-- Query 1 without using limit  
  
-- Query 2 by using limit

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.

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.

Here is the query result. 120 records returned.  

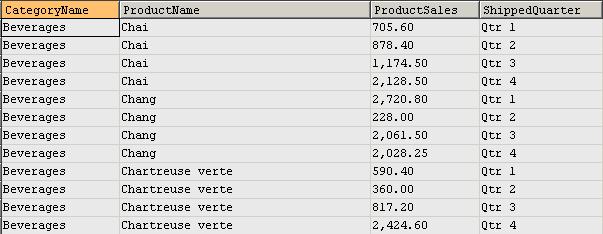

**11. Products above Average Price**

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

Here is the query result. 25 records returned.  

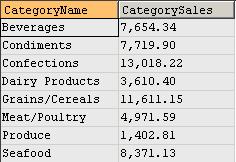

**12. Product Sales for 1997**

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

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.

Here is the query result. 8 records returned.  


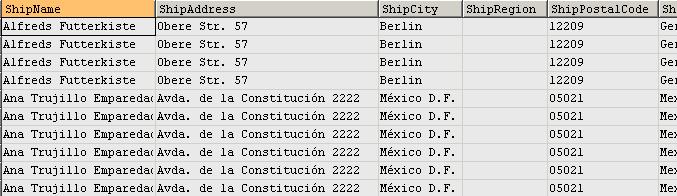
**14. Quarterly Orders by Product**

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.

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.

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.

Here is the query result. 21 records returned.  
