* 1. Write SELECT INTO statements to create two test tables named VendorCopy and InvoiceCopy that are complete copies of the Vendors and Invoices tables. If VendorCopy and InvoiceCopy already exist, first code two DROP TABLE statements to delete them.
  2. Write an INSERT statement that adds a row to the InvoiceCopy table with the following values:

VendorID: 32 InvoiceTotal: $434.58 TermsID: 2

InvoiceNumber: AX-014-027 PaymentTotal: $0.00 InvoiceDueDate: 11/8/02 InvoiceDate: 10/21/02 CreditTotal: $0.00 PaymentDate: null

* 1. Write an INSERT statement that adds a row to the VendorCopy table for each

non-California vendor in the Vendors table. (This will result in duplicate vendors in the VendorCopy table.)

* 1. Write an UPDATE statement that modifies the VendorCopy table. Change the default account number to 403 for each vendor that has a default account number of 400.
  2. Write an UPDATE statement that modifies the InvoiceCopy table. Change the PaymentDate to today’s date and the PaymentTotal to the balance due for each invoice with a balance due. Set today’s date with a literal date string, or use the GETDATE() function.
  3. Write an UPDATE statement that modifies the InvoiceCopy table. Change TermsID to 2 for each invoice that’s from a vendor with a DefaultTermsID of 2. Use a subquery.
  4. Solve exercise 6 using a join rather than a subquery.
  5. Write a DELETE statement that deletes all vendors in the state of Minnesota from the VendorCopy table.
  6. Write a DELETE statement for the VendorCopy table. Delete the vendors that are located in states from which no vendor has ever sent an invoice.

Hint: Use a subquery coded with “SELECT DISTINCT VendorState” introduced with the NOT IN operator.

* 1. Write a SELECT statement that returns four columns based on the InvoiceTotal column of the Invoices table:
  + Use the CAST function to return the first column as data type decimal with 2 digits to the right of the decimal point.
  + Use CAST to return the second column as a varchar.
  + Use the CONVERT function to return the third column as the same data type as the first column.
  + Use CONVERT to return the fourth column as a varchar, using style 1.
  1. Write a SELECT statement that returns four columns based on the InvoiceDate column of the Invoices table:
  + Use the CAST function to return the first column as data type varchar.
  + Use the CONVERT function to return the second and third columns as a varchar, using style 1 and style 10, respectively.
  + Use the CAST function to return the fourth column as data type real.
  1. Write a SELECT statement that returns two columns based on the Vendors table. The first column, Contact, is the vendor contact name in this format: first name followed by last initial (for example, “John S.”) The second column, Phone, is the VendorPhone column without the area code. Only return rows for those vendors in the 559 area code. Sort the result set by first name, then last name.
  2. Write a SELECT statement that returns the InvoiceNumber and balance due for every invoice with a non-zero balance and an InvoiceDueDate that’s less than 30 days from today.
  3. Modify the search expression for InvoiceDueDate from the solution for exercise 4. Rather than 30 days from today, return invoices due before the last day of the current month.
  4. Write a summary query WITH CUBE that returns LineItemSum (which is the sum of InvoiceLineItemAmount) grouped by Account (an alias for AccountDescription) and State (an alias for VendorState). Use the CASE and GROUPING function to substitute the literal value “\*ALL\*” for the summary rows with null values.