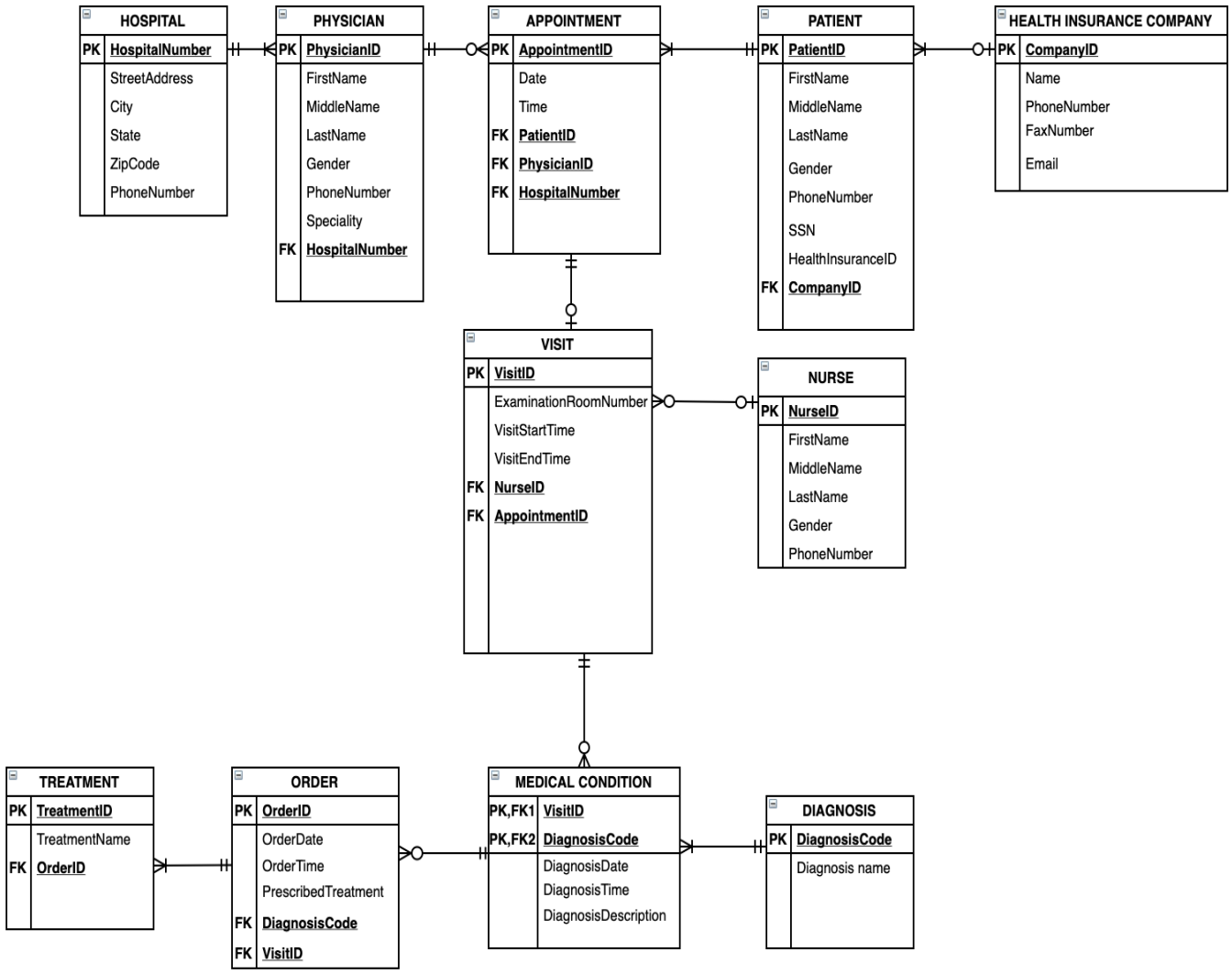
Database Management

&

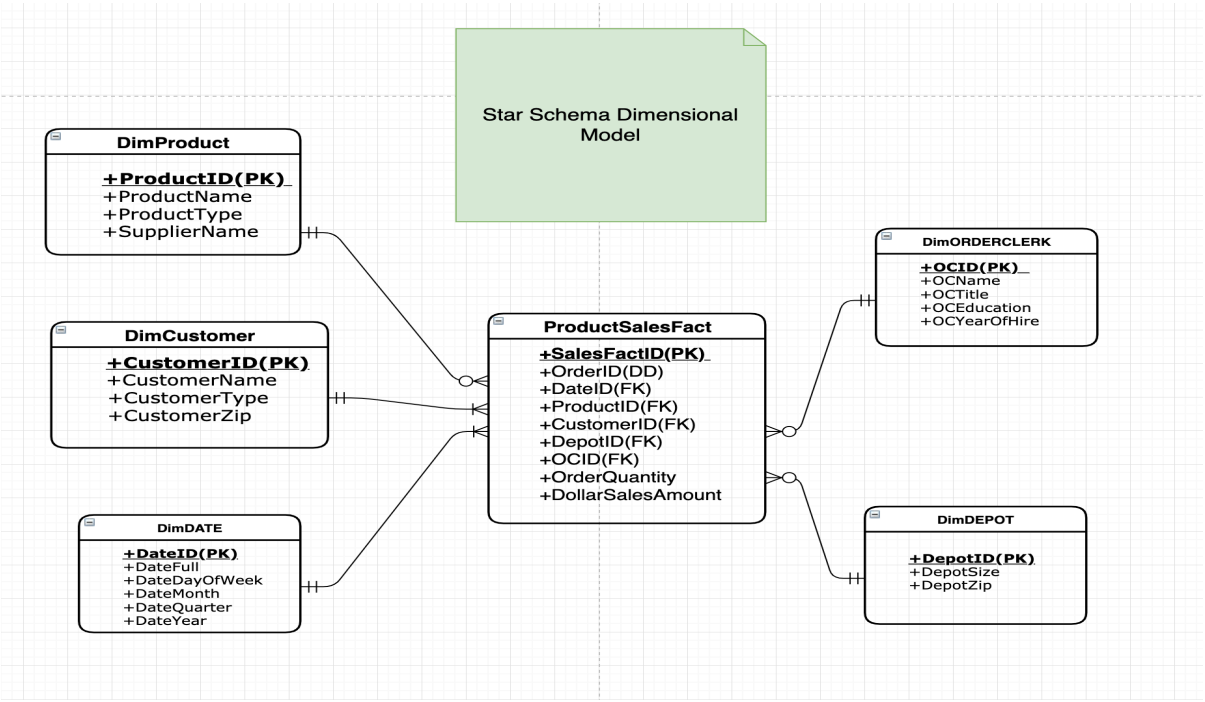
Data Warehousing

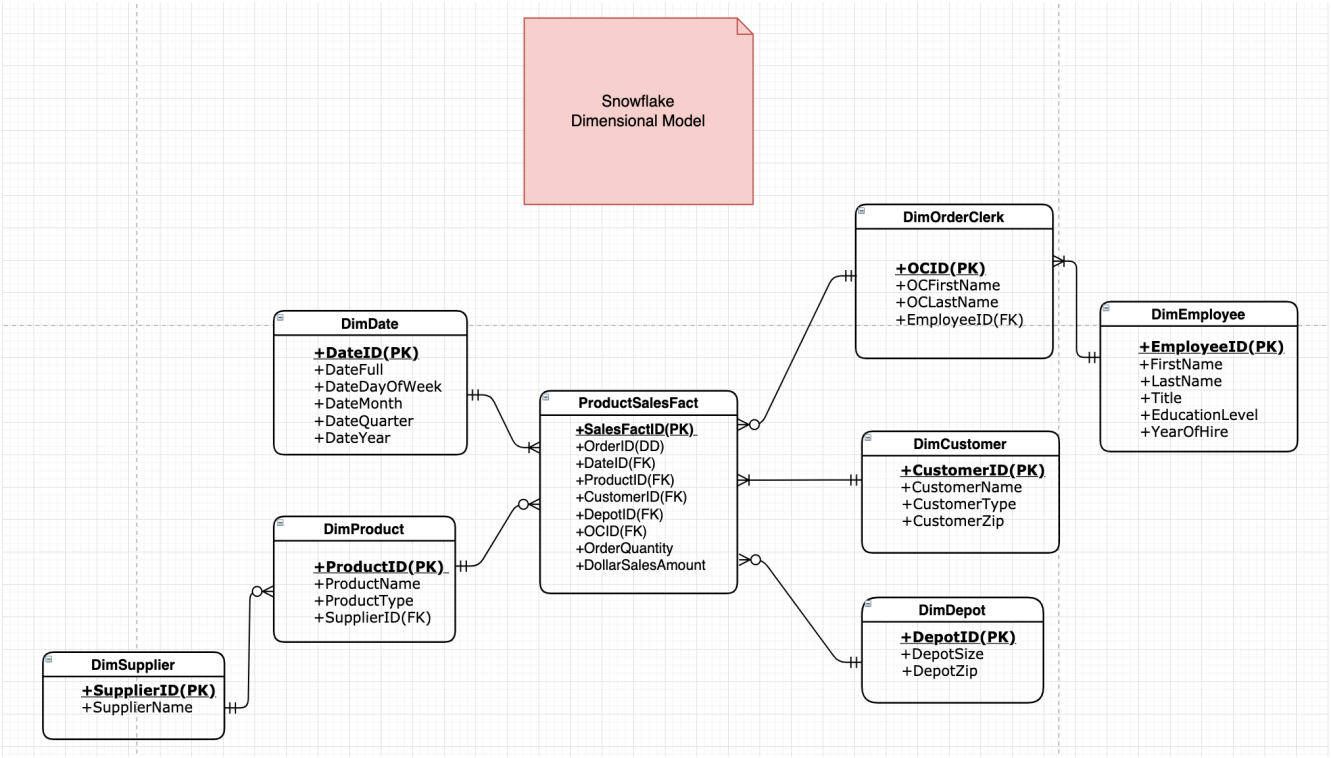
Entity Relationship Model for Hospital Management:



Texas Tire & Battery Inc. is a wholesaler of automotive products. Company executives have recruited your team as a consultant to design and document a data warehouse to support decision making. Company has a chain of depots from which it fulfills customers' orders. The data warehouse design should enable analysis of product sales in terms of dollar amounts and order quantities.

Dimensional Model for Texas Tire & Battery Inc:





Demonstrates filters, sorting, aggregation, joins:

1. List the *Store*, *City*, *State*, and *Zip code* for *stores located in California* and *Texas*. Display result-set in *ascending* order of states, and for each state in *ascending* order of city names. (**Note** – Please refer to theStore table).

```
SELECT DISTINCT STORE, CITY, STATE, ZIP_CODE
FROM STORE
WHERE STATE IN ('CA','TX')
ORDER BY STATE ASC, CITY ASC;
```

2. List the *Customer ID (Cust_ID)*, *Transaction amount (Tran_Amt)*, *Transaction date (Tran_Date)*, *City*, *State*, and *Zip code (Zip_Code)* for the *largest purchase transaction (Tran_Type)* by *customer from Texas(State)*. Information about the purchase transaction should be *displayed once.* (**Note** – Please refer to the Customer and Transact tables).

```
SELECT DISTINCT TOP 1 WITH TIES
CUSTOMER.CUST_ID AS 'Customer ID',
TRAN_AMT AS 'Transaction amount',
TRAN_DATE AS 'Transaction date',
City,
State,
ZIP_CODE AS 'Zip code'
FROM CUSTOMER
JOIN TRANSACT
ON CUSTOMER.CUST_ID = TRANSACT.CUST_ID
WHERE STATE = 'TX'
AND TRAN_TYPE = 'P'
ORDER BY TRAN_AMT DESC;
```

3. Display with appropriate column heading the *total purchase amount (Tran_Type)* in *year 2015 (Tran_Date)* at *stores located in Texas (State)*. (**Note** – Please refer to the Transact and Store tables).

```
SELECT STORE.STORE, SUM(TRAN_AMT) AS 'TOTAL PURCHASE AMOUNT'
FROM STORE
JOIN TRANSACT
ON TRANSACT.STORE = STORE.STORE
WHERE STATE = 'TX'
AND YEAR(TRAN_DATE) = '2015'
AND TRAN_TYPE = 'P'
GROUP BY STORE.STORE
ORDER BY STORE.STORE;
```

4. For *stores located in Texas (State)*, what is the *total purchase amount (Tran_Type)* for *each year (Tran_Date)*. In the result-set, include two columns: *year* and *total dollar amount of purchase for eachyear*. (**Note** – Please refer to the Transact and Store tables).

```
SELECT
YEAR(TRAN_DATE) AS 'Year',
SUM(TRAN_AMT) AS 'Total dollar amount of purchase'
FROM STORE
JOIN TRANSACT
ON TRANSACT.STORE = STORE.STORE
WHERE STATE = 'TX'
AND TRAN_TYPE = 'P'
GROUP BY YEAR(TRAN_DATE)
 ORDER BY Year ASC;
```

5. List the *Product SKU (SKU)*, *Department description (Dept_Desc)*, *DeptDec_Desc*, *Color*, *Retail price (Retail)*, and *Size (SKU_Size) for navy (Color) career (DeptDec_Desc) sweaters (Classification)* with *retailprice (Retail) less than or equal to $100* from *Jones Signature* or *Daniel Cremieux (Dept_Desc)*. Display result-set in *descending* order of *Department Description (Dept_Desc)*, and for each Department Description in *ascending* order of *Size* (SKU_SIZE). Information for each Product SKU should be *displayed once*. (**Note** – Please refer to the Department, SKU, and SKU_Store tables).

```
SELECT DISTINCT
SKU.SKU AS 'Product SKU',
DEPT_DESC AS 'Department description',
DeptDec_Desc,
Color,
SKU_STORE.RETAIL AS 'Retail price',
SKU_SIZE AS 'Size'
FROM DEPARTMENT
JOIN SKU
ON DEPARTMENT.DEPT = SKU.DEPT
JOIN SKU_STORE
ON SKU.SKU = SKU_STORE.SKU
WHERE DEPT_DESC IN ('Jones Signature','Daniel Cremieux')
AND COLOR LIKE '%NAVY%'
AND DEPTDEC_DESC LIKE '%CAREER%'
AND CLASSIFICATION LIKE '%SWEATER%'
AND SKU_STORE.RETAIL <=100
ORDER BY DEPT_DESC DESC, SKU_SIZE ASC;
```

6. In terms of the *Dept_Desc*, which *perfume/cologne (Classification)* generated the *highest total dollar sales (Tran_Amt)*. Display the *Dept_Desc* and *total dollar sales amount*. (**Note** – Please refer to the Department, SKU, and Transact tables).

```
SELECT TOP 1 WITH TIES
Dept_Desc,
SUM(TRAN_AMT) AS 'Dollar sales amount'
FROM DEPARTMENT
JOIN SKU
ON DEPARTMENT.DEPT = SKU.DEPT
JOIN TRANSACT
ON SKU.ITEM_ID = TRANSACT.ITEM_ID
WHERE SKU.CLASSIFICATION = 'PERFUME/COLOGNE'
GROUP BY Dept_Desc
ORDER BY SUM(TRAN_AMT) DESC;
```

7. List the *Customer ID (Cust_ID)*, *City*, *State*, and *Zip Code (Zip_Code)* for customers from *Texas (State)* who have *total purchase of $15,000 or more (total of Tran_Amt)*. Display these customers in *descending*order of *total purchase amount*. (**Note** – Please refer to the Customer and Transact tables).

```
SELECT DISTINCT
CUSTOMER.CUST_ID AS 'Customer ID'
, City
, State
, ZIP_CODE AS 'Zip code'
, SUM(TRAN_AMT) AS TOTAL_PURCHASE
FROM CUSTOMER
INNER JOIN TRANSACT
ON CUSTOMER.CUST_ID = TRANSACT.CUST_ID
WHERE STATE LIKE 'TX'
GROUP BY CUSTOMER.CUST_ID, CITY, STATE, ZIP_CODE
HAVING SUM(TRAN_AMT) >= 15000
ORDER BY TOTAL_PURCHASE DESC;
```

8. List the *Department Code (Dept)*, *Department Description (Dept_Desc)*, *DeptDec_Desc*, *DeptCent*, and*Deptcent_Desc* for departments with *no items* in the *SKU* table. *Exclude* from the list, departments with *Lease* in the *Deptcent_Desc* column. (**Note** – Please refer to the Department and SKU tables).

```
SELECT DEPARTMENT.DEPT AS 'Department Code',
DEPT_DESC AS 'Department Description',
DeptDec_Desc,
DeptCent,
Deptcent_Desc
FROM DEPARTMENT
LEFT JOIN SKU
ON DEPARTMENT.DEPT = SKU.DEPT
WHERE SKU.DEPT IS NULL
AND DeptCent_Desc <>'Lease';
```

9. For each store, list the *Store*, *City*, *State*, *Zip Code (Zip_Code)*, and *number of items (SKU) with retail price(Retail) $500 or more.* Display result-set in *descending* order of the number of items with retail price $500 or more. (**Note** – Please refer to the Store and SKU_Store tables).

```
SELECT DISTINCT
STORE.Store, City, State,
ZIP_CODE AS 'Zip code',
COUNT(DISTINCT SKU) AS 'Number of Items'
FROM STORE
INNER JOIN SKU_STORE
ON STORE.STORE = SKU_STORE.STORE
WHERE RETAIL >=500
GROUP BY STORE.STORE, CITY, STATE, ZIP_CODE
ORDER BY 'Number of Items' DESC;
```

Demonstrates sub queries:

10. List *CustomerID (Cust_ID)*, *City*, *State*, *Zip code (Zip_Code)*, and *Dept_Desc* for customers from Texas who have purchased *both Chanel and Armani (Dept_Desc) perfume/cologne (Classification)*. Display eachcombination of customer and perfumes purchased *once*. Sort result-set in *ascending* order of city, and for each city in *ascending* order of zip code. (**Note** – Please refer to the Department, SKU, Transact, and Customer tables. For Dept_Desc condition, use the LIKE operator with '%Chanel%' and '%Armani%'.)

```
SELECT DISTINCT SELECTED_CUSTOMERS.CUST_ID, SELECTED_CUSTOMERS.City,
SELECTED_CUSTOMERS.State, SELECTED_CUSTOMERS.ZIP_CODE,
DEPARTMENT.DEPT_DESC
FROM
(
SELECT DISTINCT CUSTOMER.CUST_ID,
CUSTOMER.City, CUSTOMER.State, CUSTOMER.ZIP_CODE
FROM CUSTOMER
WHERE CUSTOMER.CUST_ID IN
(
SELECT DISTINCT CUSTOMER.CUST_ID
FROM DEPARTMENT
JOIN SKU
ON DEPARTMENT.DEPT = SKU.DEPT
JOIN TRANSACT
ON SKU.SKU = TRANSACT.SKU
JOIN CUSTOMER
ON TRANSACT.CUST_ID = CUSTOMER.CUST_ID
WHERE (DEPT_DESC LIKE '%Chanel%')
AND SKU.CLASSIFICATION = 'Perfume/Cologne'
AND STATE LIKE 'TX'
)
AND CUSTOMER.CUST_ID IN
(
SELECT DISTINCT CUSTOMER.CUST_ID
FROM DEPARTMENT
JOIN SKU
ON DEPARTMENT.DEPT = SKU.DEPT
JOIN TRANSACT
ON SKU.SKU = TRANSACT.SKU
JOIN CUSTOMER
ON TRANSACT.CUST_ID = CUSTOMER.CUST_ID
WHERE (DEPT_DESC LIKE '%Armani%')
AND SKU.CLASSIFICATION = 'Perfume/Cologne'
AND STATE LIKE 'TX'
)
) SELECTED_CUSTOMERS
LEFT JOIN TRANSACT
ON TRANSACT.CUST_ID = SELECTED_CUSTOMERS.CUST_ID
LEFT JOIN SKU
ON TRANSACT.ITEM_ID = SKU.ITEM_ID
LEFT JOIN DEPARTMENT
ON DEPARTMENT.DEPT = SKU.DEPT
WHERE (DEPT_DESC LIKE '%Chanel%' OR Dept_Desc LIKE '%Armani%')
ORDER BY CITY ASC, Zip_code ASC;
```

11. Write a SELECT statement to list the Account Number and Account Description for General Ledger Accounts that have no invoice line items posted yet. Display the AccountNo and AccountDescription in ascending order of AccountDescription. (Note – Please refer to the GLAccounts and InvoiceLineItems tables).

```
select AccountNo, AccountDescription
from GLAccounts
where GLAccounts.AccountNo not in (select InvoiceLineItems.AccountNo from InvoiceLineItems
where GLAccounts.AccountNo = InvoiceLineItems.AccountNo)
            order by AccountDescription asc;
```

12. Write a SELECT statement to display InvoiceID, InvoiceDate, InvoiceTotal, TotalCredits (CreditTotal + PaymentTotal), TermsID, TermsDescription, and InvoiceLineItemAmount for invoices with TermsID 3 that are not fully paid (InvoiceTotal – PaymentTotal – CreditTotal > 0) and have InvoiceTotal greater than or equal to $500. Display the result-set in descending order of the InvoiceTotal values. (Note – Please refer to the Invoices, Terms, and InvoiceLineItems tables).

```
select Invoices.InvoiceID,InvoiceDate,InvoiceTotal, (PaymentTotal+CreditTotal) as TotalCredits, Terms.TermsID,
   TermsDescription, InvoiceLineItems.InvoiceLineItemAmount
from Invoices
inner join InvoiceLineItems on Invoices.InvoiceID = InvoiceLineItems.InvoiceID
inner join Terms on Invoices.TermsID = Terms.TermsID
where Terms.TermsID = 3
and (InvoiceTotal - PaymentTotal - CreditTotal)>0
and InvoiceTotal >= 500
order by InvoiceTotal desc;
```

13. Display the VendorID, VendorName, InvoiceID, InvoiceDate and InvoiceTotal for invoice(s) with invoice total amount greater than the largest invoice total amount for invoices from vendors located in California. Display the result-set in descending order of InvoiceTotal values. (Note – Please refer to the Vendors and Invoices tables).

```
select Vendors.VendorID, VendorName, InvoiceID, InvoiceDate , InvoiceTotal from
Vendors inner join Invoices on Vendors.VendorID = Invoices.VendorID
where Invoices.InvoiceTotal > (select Max(Invoices.InvoiceTotal) from
Vendors inner join Invoices on Vendors.VendorID = Invoices.VendorID AND Vendors.VendorState = 'CA')
order by InvoiceTotal desc;
```

14. Write a SELECT statement to display the TermsID, TermsDescription, Total of Invoice Amounts (i.e., total of invoice total values), Number of Invoices, and Average Invoice Total for each payment term that has ten or more invoices. Display the result-set in ascending order of Average Invoice Total values. In the result-set, provide appropriate column names for the total of invoice amounts, number of invoices, and average invoice total columns. (Note – Please refer to the Terms and Invoices tables).

```
select Terms.TermsID, TermsDescription,sum(InvoiceTotal) as 'Total of Invoice Amounts', count (*) as 'NumberOfInvoices',
avg(InvoiceTotal) as 'AverageInvoiceTotal'
from Invoices inner join Terms on Invoices.TermsID = Terms.TermsID
group by Terms.TermsID,TermsDescription
having count(*) >= 10
order by 'AverageInvoiceTotal';
```

15. Write a SELECT statement to display the VendorID and VendorName for vendors with invoices in February 2012 and March 2012. Display the result-set in ascending order of VendorName values. In the result-set each vendor should be displayed once. (Note – Please refer to the Vendors and Invoices tables).

```
select distinct Vendors.VendorID, Vendors.VendorName
from Vendors inner join Invoices on Vendors.VendorId = Invoices.VendorID
where (year (InvoiceDate) = '2012' and month(InvoiceDate) = '02')
or (year (InvoiceDate) = '2012' and   month(InvoiceDate) = '03')
And VENDORs.vendorId in (select distinct vendors.vendorID
                         from Vendors inner join Invoices on Vendors.VendorId = Invoices.VendorID
                         where (year (InvoiceDate) = '2012' and month(InvoiceDate) = '02')
                         And
VENDORs.vendorId in  (select distinct vendors.vendorID
                         from Vendors inner join Invoices on Vendors.VendorId = Invoices.VendorID
                         where (year (InvoiceDate) = '2012' and month(InvoiceDate) = '03')))
order by VendorName asc
```