Data Mining and Statistical Analysis Using SQL

ROBERT P. TRUEBLOOD AND JOHN N. LOVETT, JR.

Apress™

ISBN (pbk): 1-893115-54-2

Printed and bound in the United States of America 12345678910

Editorial Directors: Dan Appleman, Gary Cornell, Jason Gilmore, Karen Watterson

Technical Editor: Alexander Werner

Project Manager, Copy Editor, and Production Editor: Anne Friedman

Compositor: Susan Glinert

Cartoonist: John N. Lovett, Jr.

Artist: Tony Jonick

Proofreaders: Carol Burbo, Doris Wong

Indexer: Carol Burbo

Cover Designer: Karl Miyajima

Marketing Manager: Stephanie Rodriguez

Distributed to the book trade in the United States by Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY, 10010

and outside the United States by Springer-Verlag GmbH & Co. KG, Tiergartenstr. 17, 69112 Heidelberg, Germany

In the United States, phone 1-800-SPRINGER; orders@springer-ny.com; http://www.springer-ny.com

Outside the United States, contact orders@springer.de; http://www.springer.de; fax +49 6221 345229

For information on translations, please contact Apress directly at 901 Grayson Street, Suite 204, Berkeley, CA, 94710 Phone: 510-549-5938; Fax: 510-549-5939; info@apress.com; http://www.apress.com

CHAPTER 3

Goodness of Fit

Poor Fit

THE AIRPORT LIMOUSINE EXAMPLE in Chapter 2 illustrates how important the knowledge of random distribution of data and the shape of the histogram can be to drawing statistically sound conclusions. Although we didn't state it in the example, the assumption that the data followed a known statistical distribution was inherent in the calculations of the three contour radii based on the standard deviation of the data values. In this chapter, we present the technique of fitting a statistical model to a set of observed data and testing for "goodness of fit."

Once we represent a set of observed data pictorially by a histogram, and we calculate measures of central tendency and dispersion, our interest may shift to the shape of the distribution of data. By constructing the histogram, the data moved from a formless collection of numbers to a visual representation that has meaning. Frequently in statistical studies the focus is to attempt to associate the observed data values with a known statistical model. The result may reveal important information about the population from which the data were sampled. In fact, if it can be shown that a known distribution "fits" the data, we can make predictions regarding the population as a whole.

Three steps are necessary to attempt to fit a known statistical distribution to a set of observed data, assuming a histogram has already been set up. These are as follows:

1. Classify the data as variables or attributes (see Chapter 1) and study the data characteristics to determine a candidate statistical model.

2. Observe the shape of the histogram to further direct the search for a model that may fit the data.

3. After deciding on a candidate model, check to see how well the observed data fits the model by conducting a "goodness of fit" test.

The classification of the data into either variables or attributes narrows your choices for a candidate statistical model, thereby making a selection somewhat easier. Statistical distributions fall into one of these two categories, and the sections that follow present the more commonly encountered distributions of each type. Distributions have unique graphs or plots (although some are very similar), so it's helpful to compare such shapes to the histogram of the observed data. Finally, once a candidate model has been selected, we follow a procedure to determine how well the model characterizes, or "overlays," the histogram of the observed data. The degree, or goodness, of fit is tested most commonly by what is known as a chi-square test (the Greek symbol used is χ^2). The test is so named because the calculation used to compare the observed to the assumed model closely follows the ideal χ^2 statistical distribution, if the observed data and the candidate model do not differ significantly. The degree to which they deviate determines the outcome of the test.

Tests of Hypothesis

Such a test falls under a category of statistical testing known as "hypothesis testing." In the diagnostic tree in Figure 1-3, note that the second branch addresses hypothesis testing. To satisfy the pure-blood statistician who might by chance be reading this, a brief discussion of tests of hypothesis is presented. Basically, a test of hypothesis begins with a statement of the hypothesis to be tested. Makes sense so far, right? The hypothesis might be that a statistic such as the mean, calculated from a set of observed sample data, differs little from the mean of the population from which the sample was collected. The problem becomes how to define "differs little." Fortunately, folks smarter than we are years ago developed a general solution to this dilemma, and thereby acquired tenure at their respective institutions. The degree of deviation of the observed from the population (or theoretical) statistic is compared to a tabulated value based on the theoretical distribution appropriate to the test. If the calculated value exceeds the table value, the conclusion reached is that the observed statistic differs so significantly from the table's that the hypothesis test fails, and the observed data therefore does not behave the way we thought. Well, it's a bit more complicated than that.

For one thing, you must establish a degree of significance before the test is conducted. This is usually stated as a percent or decimal fraction, such as 5% (which is probably the most commonly used value). This significance level is easier to understand if viewed pictorially. If we look back at the histogram for the limousine service data in Figure 2-5, also shown in Figure 3-1, we notice that the bulk of the observations cluster around the center of the bar chart, near the mean value. The histogram is tallest in this area, representing the higher number of data values. As we move to the right or to the left of center, the heights of the bars decrease, indicating fewer and fewer observations the further we deviate from the mean. Now, we draw a smooth line near the tops of the bars on the histogram as illustrated in Figure 3-1. This curve looks much like the curve in Figure 3-2 that represents a candidate statistical model.

Of course, the percentage of area under the curve of the candidate model is 100% and represents the entire population. If we want to select a region representing only 5% of this area, one option would be to pick a point in the "upper tail" of the curve to the right of which is 5% of the area. This is illustrated in Figure 3-2. Thus, any calculation or observation that falls within this relatively small area of the distribution is considered significantly different than most of the rest of the observations. The table we mentioned earlier (available in statistics texts and in our Appendix B) is used to determine the value of the distribution to the right of which is this 5% area.

Statisticians call this type of test a "one-tailed" test, since only one tail of the distribution has been employed. However, we may in some situations want to run a "two-tailed" test. In these cases, the total significance area is usually divided equally between both tails of the curve. If we stay with 5%, that means that 2.5% falls in each tail, as shown in Figure 3-3. The test is a bit more involved, so for now let's stick with the one-tailed variety.

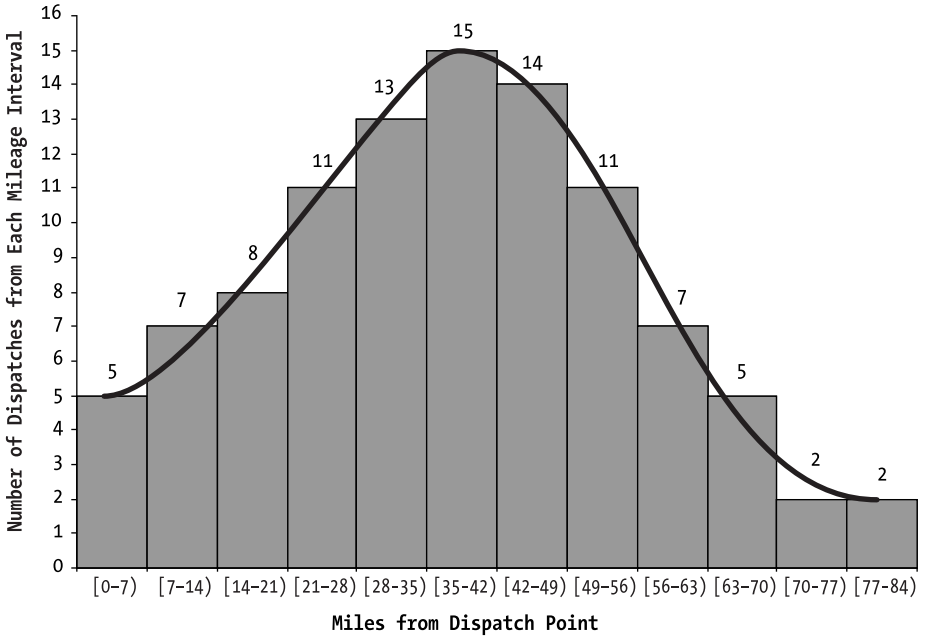

Figure 3-1. Histogram from Figure 2-5 with the bar height curve added

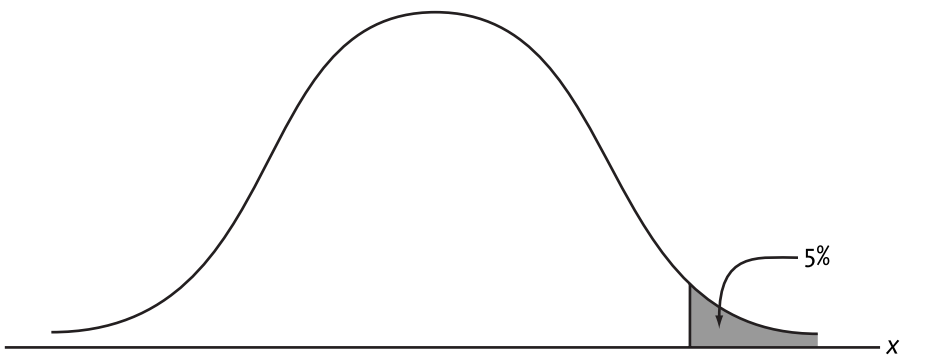

Figure 3-2. A typical one-tailed significance level for a test of hypothesis

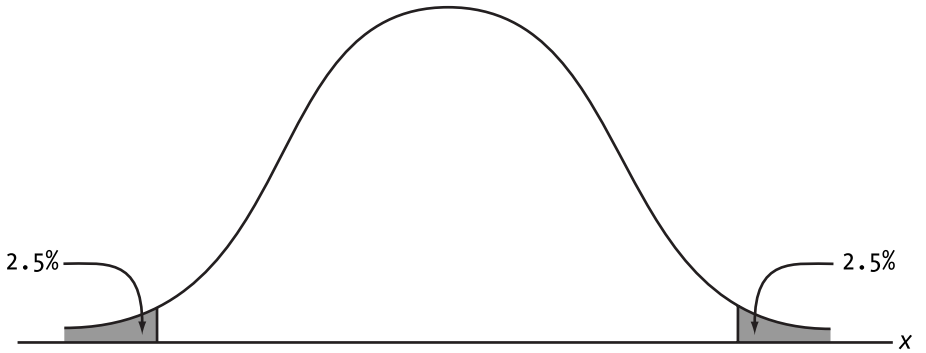

Figure 3-3. A typical two-tailed significance level for a test of hypothesis

There is one more parameter that must be used to extract the appropriate value from the table. This is the number of ***degrees of freedom*** (you know it's important if it's in bold italics). This is one of the most difficult concepts in statistics to explain to anyone. Perhaps it is easiest to present it as follows: Recall from Chapter 2 that the calculation of the standard deviation of a collection of sample values was

$$s = \sqrt{\frac{(n)\left(\sum x^2\right) - \left(\sum x\right)^2}{(n)(n-1)}}$$

An equivalent form of this expression is

$$s = \sqrt{\frac{\sum (x - \bar{x})^2}{n-1}}$$

where $\sum (x - \bar{x})^2$ is calculated by first finding the mean $\bar{x}$ of all n data values, then subtracting it from each individual data value (represented by x), squaring the results one at a time, and adding all the n squared results together. The $n-1$ in the denominator of the expression is the number of degrees of freedom for the statistic.

Suppose we want to calculate the standard deviation of four values, say 3, 5, 6, and 10. The mean $\bar{x}$ of these values is 6. To calculate the standard deviation, we subtract the mean from each of the four values, as follows:

$3 - 6 = -3 \qquad 5 - 6 = -1 \qquad 6 - 6 = 0 \qquad 10 - 6 = 4$

Now square each of these results:

$$(-3)^2 = 9 \qquad (-1)^2 = 1 \qquad (0)^1 = 0 \qquad (4)^2 = 16$$

Then sum the four values, yielding 26. Divide by $n-1=3$, take the square root of the result, and the standard deviation is equal to 2.94.

Now let's explain the degrees of freedom. Notice that the four values found by subtracting the mean from each observation sum to zero. Therefore, you can take any three of these differences and obtain the fourth with this knowledge. For example, knowing the first three differences are -3, -1, and 0, and the sum of all four is 0, the fourth difference has to be 4. Since any three of the four differences yield the fourth, there are three "free" variables, hence three degrees of freedom. For the sample standard deviation calculation, the number of degrees of freedom is always equal to one less than the number of observations (i.e., $n-1$). Another way of stating this is that, since the sample statistic $\bar{x}$ was used to estimate the standard deviation, and its value was obtained from the sample data itself, this fact reduced the number of degrees of freedom by one. Clear as mud? Good, now we can proceed from shaky ground to the quicksand.

John's Jewels
The Degrees of Freedom of Flightless Birds

In Chapter 7 we revisit the concept of degrees of freedom when we discuss experimental design analyses. At that point we state that the number of degrees of freedom is basically the number of independent variables in a problem less the number of constraints. For example, consider a bird in an aviary. If the bird can fly, it can move in any direction of length, width, and height of the aviary, so it has three degrees of freedom in which to travel. However, think about a flightless bird in the aviary. It can basically only move across the floor, in two directions (length and width). It is constrained by its inability to fly. Therefore, it has only two degrees of freedom (three directions less one constraint). Maybe this makes the concept a bit easier to understand.

Goodness of Fit Test

Generally speaking, tests of hypotheses have similar characteristics. However, for specific testing, there are slight variations, and the goodness of fit test is a typical example. After the candidate distribution model is selected for the goodness of fit test, it is necessary to

use this model and one or more parameters estimated from the observed data (such as the mean and standard deviation) to calculate values predicted by the model. These values are compared to the observed values in intervals of the histogram. The comparison involves calculating a statistic that is the basis for the chi-square test. This will (perhaps) become clear when we present the following examples.

Fitting a Normal Distribution to Observed Data

The histogram of limousine service data presented in Figure 3-1 has a shape that we very commonly encounter in real-world situations, where the values are measured quantities (variables data). If we trace a smooth curve through the tops of the bars on the histogram, the resulting shape looks like a bell, so it is known as a bell curve. Statisticians prefer a more academic term, so the curve is said to represent a *normal* distribution. The normal statistical distribution is basic to an understanding of data mining and analysis. It is a type of *continuous* distribution, which means that the data whose histogram is bell-shaped assume continuous, or variables, values. In the case of the limousine service data, the observations were measured in miles from a point. The precision of these measurements was limited only to the method of measurement and the needs of the analyst. In this example, the results were simply rounded to the nearest mile, but in general they could have been measured to tenths or hundredths of a mile, or even more precise values. For the purpose of the analysis, this was not necessary, but the point is that the values could potentially assume any number within the observed range between 2 and 83 miles. This is why the data are termed *continuous*.

A continuous statistical distribution like the normal is represented by what is known as a *density* function. This is simply an algebraic expression that defines the shape of the distribution, and may be used to calculate areas under the normal curve between points on the horizontal axis. These points represent the various values that the range of the distribution might be expected to assume. In the case of an ideal normal distribution, the values range, in general, over the entire set of real numbers (both positive and negative). For most practical data sets, the numbers range from zero to (positive) infinity. If we integrate the normal density function between any two desired values within its range, it is possible to determine the area under the curve in that interval. This is especially useful in testing for goodness of fit. Fortunately, it is not necessary to actually integrate the horrible looking functional expression to obtain these areas (in fact, it is impossible to integrate it exactly!). Tables have been developed to accomplish this. For those stalwart few who wish to see the actual normal density function expression, cast your eyes below.

$$\frac{1}{\sqrt{2\pi\sigma}} e^{-(1/2)[(x-\mu)/\sigma]^2}$$

In the normal density function, μ denotes the mean, σ denotes the standard deviation, and x is the variable (e.g., miles traveled). Often μ and σ are unknown and we estimate them from the sample data by $\bar{x}$ and s, respectively. Of course π = 3.14159.

The procedure for testing to see if the limousine data actually fit a normal distribution is a little involved, so please bear with us. The first step is to calculate the mean and standard deviation of the original data. These are $\bar{x} = 36.56$ miles and $s = 19.25$ miles. Both these parameters are necessary to determine expected numbers of observations (or expected frequencies) to compare to the actual observations. The method by which this is accomplished is to calculate the area in each bar of the histogram. If the top of that bar was really determined by the curve of the (ideal) normal distribution, then multiply the result by the total number of observations (in this case 100). The results are compared to the observed frequencies interval by interval to determine the extent of deviation. However, there is a little rule that you should follow: If in any interval (or bar) of the histogram, the number of data values *expected* is less than 5, this interval is combined with one or more neighboring intervals until the total values are at least 5. The same values are then combined for the observed frequencies. This rule will be demonstrated when we calculate the expected frequencies a little later. Table 3-1 summarizes the data from the histogram in Figure 3-1.

Table 3-1. Number of Limousine Pick-Up Points Falling Within Each Mileage Interval from the Dispatching Location at the Airport

HISTOGRAM INTERVAL (MILES)	OBSERVED FREQUENCY (*O*)
[0 - 7)	5
[7 - 14)	7
[14 - 21)	8
[21 - 28)	11
[28 - 35)	13
[35 - 42)	15
[42 - 49)	14
[49 - 56)	11
[56 - 63)	7
[63 – 70)	5
[70 – 77)	2
[77 – 84)	2

SQL/Query

The data shown in Table 3-1 can be generated by executing a set of SQL queries. In our first query we create an initial version of Table 3-1 by forming the intervals of width 7 and by giving each interval an identification number. Query 3_1 accomplishes the task. Just a reminder: Notice that the table in Query 3_1 is named Table 3_1 rather than Table 3-1. This is because most implementations of SQL do not allow a hyphen in a table or field name. The result of Query 3_1 is illustrated in Table 3-2.

Query 3_1:

```
SELECT Fix(([Miles])/7) AS [Interval],
(Fix(([Miles])/7))*7 AS LowEnd,
(Fix(([Miles])/7+1))*7 AS HiEnd,
Count([Limo Miles].Miles) AS CountOfMiles
INTO [Table 3_1]
FROM [Limo Miles]
GROUP BY Fix(([Miles])/7), (Fix(([Miles])/7))*7, (Fix(([Miles])/7+1))*7
ORDER BY Fix(([Miles])/7);T
```

Table 3-2. Result of Query 3_1

INTERVAL	LOWEND	HIEND	COUNTOFMILES
0	0	7	5
1	7	14	7
2	14	21	8
3	21	28	11
4	28	35	13
5	35	42	15
6	42	49	14
7	49	56	11
8	56	63	7
9	63	70	5
10	70	77	2
11	77	84	2

To calculate the frequencies we would expect if the data were exactly normally distributed, we must either employ the table of areas under the normal curve found in Appendix B

or use the Visual Basic code in Appendix D. Most forms of this table give the area under the normal curve from negative infinity up to the desired value. The normal curve used in the table, however, is called a *standardized* normal distribution, so that it can be used for any normal curve, so long as a transformation of variables is accomplished. The standardized normal curve has its mean at zero and standard deviation equal to one. For our example, the transformation is accomplished described in the text that follows:

The first interval on the histogram is from 0 to (but not including) 7 miles. If we overlay a normal distribution onto the histogram, a portion of the curve actually falls in the negative range to the left of zero. In practice we know we can't have any distances less than zero, but we still need to accommodate this very small (theoretical) tail region when we determine the area under the normal curve up to 7 miles. This area is illustrated in Figure 3-4. To use the normal table to determine this area, we need to calculate the following expression, which represents the transformation of variables:

$$\frac{x-\bar{x}}{s}$$

where x is the right-hand endpoint of the histogram interval, $\bar{x}$ is the mean of the data, and s is the standard deviation. For the first interval, this expression is equal to

$$\frac{7-36.56}{19.25}=-1.536$$

This value is called the ***normal variate***, and is used to find the desired area in the normal table. If we find -1.53 in the table and interpolate for the third decimal place (which usually isn't necessary), the corresponding area is 0.0623, or 6.23%. This means that 6.23% of the area under the normal curve of mileage is to the left of 7 miles. For 100 observations, this means 6.23 is an expected frequency of occurrence. As a comparison, in the actual data there were 5 customer calls less than 7 miles from the limousine headquarters, or 5% of the sample data.

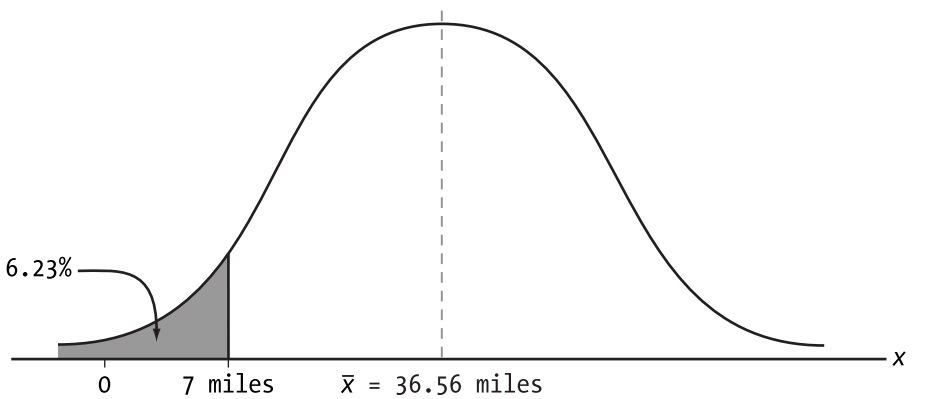

Figure 3-4. Theoretical normal distribution of limousine customer pick-ups, showing the area under the curve to the left of 7 miles

The next histogram interval is between 7 and 14, so we need to calculate the area under the normal curve between these two values. To do this using the normal table, we would first find the area to the left of 14 by using the procedure outlined above, then subtract from it the area to the left of 7 (already calculated). The difference would be the area between 7 and 14. Therefore, the following calculation is first performed:

$$\frac{14 - 36.56}{19.25} = -1.172$$

The area to the left of this value in the normal table is 0.1206, or 12.06%. By subtracting 6.23% from this area, we obtain the desired area between 7 and 14. This is 5.83%, or 5.83 occurrences for 100 values. Continuing in this manner, Table 3-3 is generated. The last expected frequency (1.782) may be found by subtracting the sum of all previous expected frequencies from 100 (in this case).

Table 3-3. Observed and Expected Frequencies of Occurrence of Call Distances (in Miles) Based on the Sample of 100 Records of the Limousine Service

HISTOGRAM INTERVAL (MILES)	OBSERVED FREQUENCY (*O*)	EXPECTED FREQUENCY (*E*)
< 7	5	6.227
[7 – 14)	7	5.833
[14 – 21)	8	8.895
[21 – 28)	11	11.861
[28 – 35)	13	13.956
[35 – 42)	15	14.369
[42 – 49)	14	12.945
[49 – 56)	11	10.289
[56 – 63)	7	7.153
[63 – 70)	5	4.353
[70 – 77)	2	2.337
≥ 77	2	1.782

SQL/Query

Now let us see how we can generate Table 3-3 using some SQL queries on Table 3-1 along with the standard normal table, StdNormal, described in Appendix B. First, we use Query 3_2 to calculate the sample mean and standard deviation. Next we execute Query 3_3 to create a seven column temporary work table called TempWork. The first column of TempWork is an interval identification number that is used as identify each interval in sequence. The second and third columns correspond to the information in Table 3-1. The fourth and fifth columns are the sample mean and standard deviation. The sixth column is the normal variate, $(x - \bar{x})/s$. The last column is the value from the StdNormal table for the normal variate. When we execute Query 3_3, it populates the first six columns of TempWork table and temporarily fills the last column with zeros. Notice that Query 3_3 extracts data from two different sources (i.e., Table 3_1 and Query 3_2) without a join condition. Whenever the join condition is omitted, a cross product is formed between the two sources. Since the result of Query 3_2 is a single row, we are in effect appending the sample mean and standard deviation to the selected rows from Table 3_1. Also notice the use of the function "Round(0, 3)" in place of "0" for the value of StdNorm. We did this so that the data type for StdNorm would be a real number rather than an integer. Table 3-4 illustrates the contents of TempWork after executing Query 3_3. It should be pointed out that the last value (2.4648) of the normal variate in the table is not used, so acts only as a place holder.

Query 3_2:

```
SELECT Avg(Miles) AS xBar, StDev(Miles) AS s, Count([Miles]) AS n
FROM [Limo Miles];
```

Query 3_3:

```
SELECT [Table 3_1].Interval,
[Table 3_1].LowEnd, [Table 3_1].HiEnd,
[Table 3_1].CountOfMiles,
[Query 3_2].xBar, [Query 3_2].s,
([HiEnd] - [xbar]) / [s] AS Nvariate,
Round(0, 3) AS StdNorm
INTO TempWork
FROM [Query 3_2], [Table 3_1];
```

Table 3-4. Initial Contents of TempWork

INTERVAL	LOWEND	HIEND	COUNTOFMILES	XBAR	S	NVARIATE	STDNORM
0	0	7	5	36.56	19.2471	-1.5358	0
1	7	14	7	36.56	19.2471	-1.1721	0
2	14	21	8	36.56	19.2471	-0.8084	0
3	21	28	11	36.56	19.2471	-0.4447	0
4	28	35	13	36.56	19.2471	-0.0811	0
5	35	42	15	36.56	19.2471	0.2826	0
6	42	49	14	36.56	19.2471	0.6463	0
7	49	56	11	36.56	19.2471	1.0100	0
8	56	63	7	36.56	19.2471	1.3737	0
9	63	70	5	36.56	19.2471	1.7374	0
10	70	77	2	36.56	19.2471	2.1011	0
11	77	84	2	36.56	19.2471	2.4648	0

Our next task is to update the StdNorm column in TempWork. Query 3_4 accomplishes this task. Notice the WHERE clause and the use of the Round function to truncate both values to three decimal places. This assures us of an equality match. Also the StdNormal table was generated to three decimal places.

Query 3_4:

```
UPDATE TempWork, StdNormal
SET TempWork.StdNorm = StdNormal.Area
WHERE Round(Nvariate,3) = Round(StdNormal.x,3);
```

Now that we have our normal values, we need to calculate the expected frequency value for each interval. As you may have already observed, the first interval is everything less than 7, and the last interval is everything greater than or equal to 77. All the intervals between the first and last have specific beginning and ending values. Thus, to calculate the expected value, we need to write three queries: one query for the first interval, one query for all intervals between the first and last, and one query for the last interval. Before we can write the queries, we need to know the lowest interval value (i.e., 7) and the highest interval value (i.e., 77). Query 3_5 accomplishes this task by extracting the interval number from the TempWork table.

Query 3_5:

```
SELECT Min(Interval) AS Lowest, Max(Interval) AS Highest
FROM TempWork;
```

Using the lowest interval value found by Query 3_5, we write Query 3_6a to obtain the expected frequency for the first interval. Notice how the expected frequency is multiplied by the total number of data values (in this specific example,100) to convert it to a number of observations, and then rounded to three decimal places.

Query 3_6a:

```
SELECT TempWork.Interval,
TempWork.LowEnd, TempWork.HiEnd,
TempWork.CountOfMiles AS O,
Round([StdNorm] * 100, 3) AS E
INTO [Table 3_3]
FROM TempWork, [Query 3_5]
WHERE TempWork.Interval = [Lowest];
```

Our next task is to calculate the expected frequency for all of the in-between intervals. Query 3_6b performs this task. Remember that the expected frequency is determined by subtracting the normal value of the preceding interval from the normal value of the current interval. We can match these two intervals by equating the number of the current interval to its preceding interval's number plus one. To do this we need to join the TempWork table with itself. To make sure that we only work with the in-between intervals, we select only those rows from TempWork with an interval number greater than the lowest but less than the highest.

Query 3_6b:

```
INSERT INTO [Table 3_3] ([Interval], LowEnd, HiEnd, O, E)
SELECT TempWork.Interval,
TempWork.LowEnd, TempWork.HiEnd,
TempWork.CountOfMiles AS O,
Round(([TempWork].[StdNorm] - TempWork_1.StdNorm) * 100, 3) AS E
FROM TempWork, TempWork AS TempWork_1, [Query 3_5]
WHERE ((TempWork.Interval = ([TempWork_1].[Interval] + 1)
And TempWork.Interval > [Lowest]
And TempWork.Interval < [highest]));
```

To determine the expected frequency for the last interval, we need to subtract the sum of all preceding expected frequencies (E) from the number of values, n. Query 3_6c and Query 3_6d perform this task. Notice in the query the use of the Round function with which we subtract the normal value from 1, and notice in the WHERE clause how we select the interval just preceding the highest.

Query 3_6c:

```
SELECT Sum([Table 3_3].E) AS Sum_Less_Last
FROM [Table 3_3];
```

Query 3_6d:

```
INSERT INTO [Table 3_3] ([Interval], LowEnd, HiEnd, O, E)
SELECT TempWork.Interval, TempWork.LowEnd, 999 AS HiEnd,
TempWork.CountOfMiles AS O,
[n] - [Sum_Less_Last] AS E
FROM TempWork, [Query 3_5], [Query 3_6c], [Query 3_2]
WHERE TempWork.Interval = [Highest];
```

The combined result of Query 3_6a through Query 3_6d is shown in Table 3-5.

Table 3-5. Result of Query 3_6a Through Query 3_6d

INTERVAL	LOWEND	HIEND	O	E
0	0	7	5	6.227
1	7	14	7	5.833
2	14	21	8	8.895
3	21	28	11	11.861
4	28	35	13	13.956
5	35	42	15	14.369
6	42	49	14	12.945
7	49	56	11	10.289
8	56	63	7	7.153
9	63	70	5	4.353
10	70	77	2	2.337
11	77	999	2	1.782

Since we need the number of expected frequencies in each interval greater than or equal to 5, we first combine all the adjacent intervals whose combined sum is less than 5. In Query 3_7, we save these combined intervals in a table called “CombInterval.” The results of Query 3_7 are given in Table 3-6.

Query 3_7:

```
SELECT [Table 3_3].Interval,
[Table 3_3].LowEnd, [Table 3_3_1].HiEnd,
[Table 3_3].[O] + [Table 3_3_1].O AS O,
[Table 3_3].[E] + [Table 3_3_1].E AS E
INTO CombInterval
FROM [Table 3_3], [Table 3_3] AS [Table 3_3_1]
WHERE ((([Table 3_3].HiEnd) = [Table 3_3_1].[LowEnd])
AND (([Table 3_3].[E] + [Table 3_3_1].[E]) < 5));
```

Table 3-6. Result of Query 3_7

INTERVAL	LOWEND	HIEND	O	E
10	70	999	4	4.119

Those intervals (rows) that were combined in Query 3_7 are replaced with the rows in the CombInterval table. To do this we first tag for deletion those rows in Table 3_3 that participated in forming the combined intervals, by setting the interval identification number to –1 (Query 3_8). Afterward we execute Query 3_9 to delete the tagged rows (intervals) from the table. Then we run Query 3_10 to append the newly formed combined interval to Table 3_3. Repeat Query 3_7 through Query 3_10 until no more intervals can be combined.

Query 3_8:

```
UPDATE CombInterval, [Table 3_3]
SET [Table 3_3].[Interval] = -1
WHERE ((([Table 3_3].LowEnd) >= [CombInterval].[LowEnd])
AND (([Table 3_3].HiEnd) <= [CombInterval].[HiEnd]));
```

Query 3_9:

```
DELETE [Table 3_3].Interval
FROM [Table 3_3]
WHERE [Table 3_3].Interval = -1;
```

Query 3_10:

```
INSERT INTO [Table 3_3]
SELECT CombInterval.*
FROM CombInterval;
```

Next, we replace Query 3-7 with Query 3_7a and run repeatedly the query sequence Query 3_7a, Query 3_8, Query 3_9, and Query 3_10 until no more intervals are combined. Table 3-7 shows Table 3_3 after the modifications.

Query 3_7a:

```
SELECT [Table 3_3].Interval,
[Table 3_3].LowEnd, [Table 3_3_1].HiEnd,
[Table 3_3].[O] + [Table 3_3_1].O AS O,
[Table 3_3].[E] + [Table 3_3_1].E AS E
INTO CombInterval
FROM [Table 3_3], [Table 3_3] AS [Table 3_3_1]
WHERE (([Table 3_3].HiEnd = [Table 3_3_1].[LowEnd])
AND ([Table 3_3].E < 5) AND ([Table 3_3_1].E < 5));
```

Table 3-7. Table 3_3 After Combining Intervals

INTERVAL	LOWEND	HIEND	COUNTOFMILES (O)	EXPECTED FREQUENCY (E)
0	0	7	5	6.227
1	7	14	7	5.833
2	14	21	8	8.895
3	21	28	11	11.861
4	28	35	13	13.956
5	35	42	15	14.369
6	42	49	14	12.945
7	49	56	11	10.289
8	56	63	7	7.153
9	63	999	9	8.472

At this point we have combined all the adjacent intervals whose combined sum is under 5 or whose individual values are under 5. This leaves us with a table in which we still might have an interval with an expected frequency less than 5. If we combine this interval with its adjacent row, the result is an interval frequency greater than or equal to 5. Thus our next query extracts the row pairs that can be combined to yield intervals greater than 5. These combined intervals are placed in the table "RemInterval" as illustrated in Table 3-8. However, since our example does not have any intervals to combine, the entries in Table 3-8 are all zeros, but in reality the result of Query 3_11 is empty (no values).

Query 3_11:

```
SELECT [Table 3_3].Interval,
[Table 3_3].LowEnd, [Table 3_3].HiEnd,
[Table 3_3].O, [Table 3_3].E
INTO RemInterval
FROM [Table 3_3]
WHERE ((([Table 3_3].E) < 5));
```

Table 3-8. Table RemInterval

INTERVAL	LOWEND	HIEND	O	E
0	0	0	0	0

Combine this remaining interval with its adjacent interval and place the result in the table "CombInterval" which is shown in Table 3-9. Again, we show all zeros since our example produced an empty result.

Query 3_12:

```
SELECT [Table 3_1].LowEnd, RemInterval.HiEnd,
[Table 3_1].[O] + [RemInterval].[O] AS O,
[Table 3_1].[E] + [RemInterval].[E] AS E
INTO CombInterval
FROM RemInterval, [Table 3_1]
WHERE ((([Table 3_1].HiEnd) = [RemInterval].[LowEnd]));
```

Table 3-9. Table CombInterval

INTERVAL	LOWEND	HIEND	O	E
0	0	0	0	0

Repeat Query 3_8, Query 3_9, and Query 3_10 to replace those intervals, if any, that have been combined with the newly combined intervals. The result of Query 3_10 is given in Table 3-10. Notice in this case that the contents of Table 3-10 are identical to Table 3-7 because there are no more intervals to combine.

Table 3-10. Results of Query 3_10

INTERVAL	LOWEND	HIEND	COUNTOFMILES	E
0	0	7	5	6.227
1	7	14	7	5.833
2	14	21	8	8.895
3	21	28	11	11.861
4	28	35	13	13.956
5	35	42	15	14.369
6	42	49	14	12.945
7	49	56	11	10.289
8	56	63	7	7.153
9	63	999	9	8.472

Once these values are known, a chi-square (χ^2) statistic is calculated by comparing pairwise the observed (O) and expected (E) frequencies, as follows:

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$
$$= \frac{(5-6.227)^2}{6.227} + \frac{(7-5.833)^2}{5.833} + \ldots + \frac{(9-8.472)^2}{8.472}$$
$$= 0.892$$

We find the number of degrees of freedom for the chi-square statistic in this case by counting the number of terms in the above sum (which is 10), subtracting 2 (representing the number of parameters needed from the data to calculate the expected frequencies—specifically the mean and standard deviation), then subtracting 1 (to account for the use of sample data). The result is of course 7 degrees of freedom. If we would like less than a 5% chance of reaching the wrong conclusion about the goodness of fit of the normal distribution, we would choose 0.05 as our significance level for the test, and either look up the χ^2 value in the appropriate statistical table in Appendix B, Table B-2, or use the Visual Basic code in Appendix D. The table value is equal to 14.067. Obviously, the calculated χ^2 value is much less than this table value, so our conclusion is that the distance data do appear to follow a normal distribution.

SQL/Query

Writing an SQL query to calculate the χ^2 can be accomplished in a relatively straightforward manner. Query 3_13 uses the Sum function to tally measured differences between the observed and expected frequencies. Either Query 3_14a or Query 3_14b can be used to obtain the χ^2 value. Query 3_14a looks up the χ^2 value from the chi-square statistics table. Query 3_14b uses a Visual Basic function to link to Excel to obtain the χ^2 value.

Query 3_13:

```
SELECT Sum((([o] - [e]) ^ 2) / [e]) AS ChiSq
FROM [Table 3_2];
```

Query 3_14a:

```
SELECT ChiSquare.ChiSq
FROM ChiSquare
WHERE ChiSquare.Percent = 0.05
AND ChiSquare.Degress_Of_Freedom = 7;
```

Query 3_14b:

```
SELECT Chi_Sq(0.05, 7) AS [Chi Sq]
FROM [Query 3_13];
```

NOTE *In Query 3_14b the FROM clause is not used. However, SQL requires a FROM clause in all queries.*

The results of the χ^2 test may be used to lend credence to the previous calculations in Chapter 2, and support the assumption of random distribution of distances. We can also use the normal distribution to answer questions about the distances. For example, what percentage of customers could be expected to call for limousine service say, more than 40 miles from the headquarters at the airport? This question is illustrated in Figure 3-5. The solution is to find the area under the normal curve to the right of 40 miles. To do

this, we may find the area to the left of 40 and subtract the result from 100%, using the normal table. Thus

$$\frac{40-36.56}{19.25}=0.1787$$

The area to the left of this value in the normal table is 0.5709, so the shaded area in Table 3-3 is equal to 1-0.5709, or 0.4291. Therefore, 42.9% of the customers could be expected to call for service more than 40 miles from the limousine headquarters.

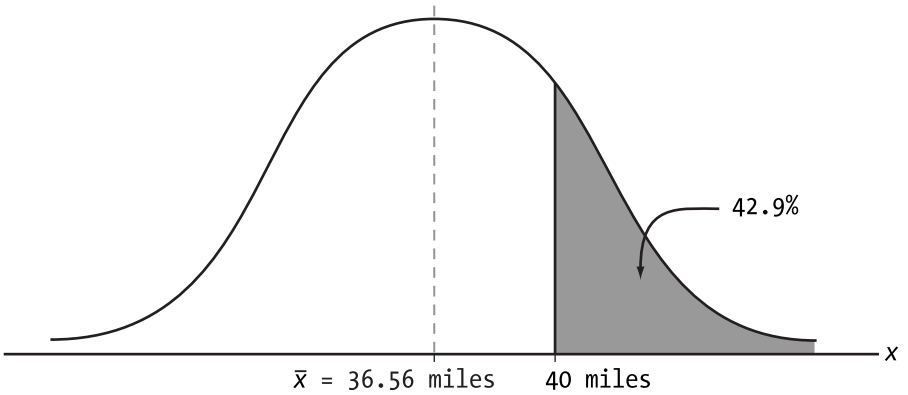

Figure 3-5. Theoretical normal distribution of limousine customer pick-ups, showing the area under the curve to the right of 40 miles

SQL/Query

An SQL query that can be used to obtain the answer to the "more than 40 miles" question is given in Query 3_15. Notice that there is no join condition, since Query 3_2 is a single row, and that the Round function is used to assure proper matching.

Query 3_15:

```
SELECT round((1.0 - [Area]) * 100, 1) AS Answer
FROM [Query 3_2], StdNormal
WHERE round([X], 3) = round((40.0 - [xBar]) / [s], 3);
```

John's Jewels
The Chance of an Accident Given the Opportunity

I recently had an opportunity to fit a statistical distribution to a set of data. We were working in the field of risk analysis and were trying to relate a certain accident risk to the opportunity for the accident to occur. An analogy would be the following: If you drove your car to work 1,000 times over a period of years, what is the likelihood that you would have an accident? Of course, the data required to develop such a model must come from past accident statistics. In our case, these data were scarce, since the types of accidents we were interested in analyzing were extremely rare. We ended up plotting the total accidents by years of occurrence, using a 7-year interval. The graph of the resulting data looked much like the curve of the gamma statistical distribution (see Appendix C), as accident occurrences became more infrequent in recent years. I tried the gamma as my model and ran a standard chi-square test on the results. They showed the gamma to be a pretty good fit for the data, so we were then able to use this information to predict the chance of a future accident.

Fitting a Poisson Distribution to Observed Data

A small retailer maintains a Web site for ordering online. The owner is particularly interested in the e-commerce sales of one high-profit item. The daily online sales of each unit of this item are tracked for 75 days, with the results recorded in Table 3-11. Each number represents the units per day in sales of this item.

Table 3-11. Number of Items Sold per Day for 75 Days

1	3	3	5	4	4	1	3	5	3	9	4	6	4	0
4	5	6	4	0	5	6	6	8	2	4	6	4	2	3
5	7	12	3	6	2	7	1	6	5	2	2	3	4	4
6	9	0	5	4	3	4	4	0	1	3	5	6	3	3
2	7	6	10	3	4	6	4	5	4	3	4	4	5	3

The owner desires to know if sales follow some random statistical distribution, based on the sample data, so that predictions may be made regarding future sales. First, a histogram is set up for the data. Since the values represent counts, the data are *discrete*,

or based on *attributes*. Since so many individual values repeat themselves, a frequency of occurrence may be determined for each value. After the counts are completed, the histogram appears in Figure 3-6.

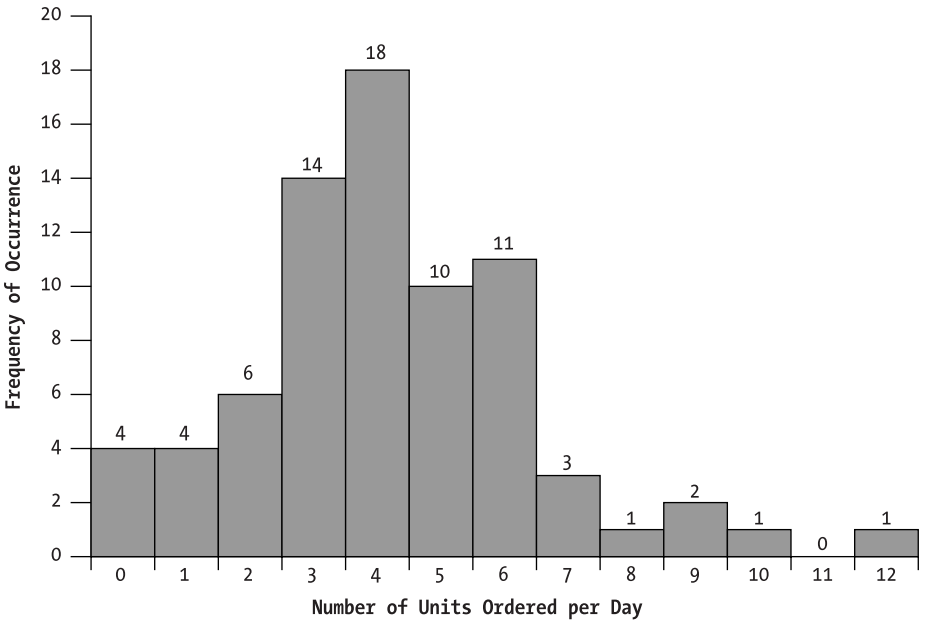

Figure 3-6. Histogram of daily sales

The shape of this histogram (rising fairly rapidly to a peak, then tapering gradually to the right) and the nature of the phenomenon being observed, point to a well-known statistical distribution called the *Poisson*. A Poisson distribution typically characterizes the occurrence of a discrete "event," such as ordering a retail item, when the number of orders may be counted. However, the "nonoccurrence" of the event makes no sense (i.e., the number of people who did not order on a given day is unknown and irrelevant). The Poisson also commonly characterizes arrival and departure phenomena, and the random spatial distribution of observations such as the number of trees per acre in a forest, or prairie dog colonies per square mile in a wildlife habitat.

Poisson Sets the Groundwork for Horse Kick Analysis

The great mathematician Simeon Denis Poisson was born in 1781 in Pithiviers, France. He was first forced to study medicine, but later became a student of the legendary mathematicians Laplace and Lagrange. During his lifetime, he excelled in applications of mathematics to astronomy, electricity and magnetism, elasticity, differential equations, potential theory, and mechanics. He published between 300 and 400 mathematical works. One of the most famous was entitled *Recherches sur la probabilite des jugements…* (1837), in which Poisson first introduced to the world the statistical distribution that still bears his name. Legend has it that one of the first practical applications of the Poisson distribution was by von Bortkewitch in 1898 (some sources say 1880). He actually employed the distribution to analyze the deaths due to horse kicks in the Prussian army. He found that, among 10 army corps over a 20-year period, 122 men had died from being kicked by their horses. When the number of such deaths per year is plotted against a theoretical Poisson distribution with mean 0.61 deaths per corps per year (the actual average), they coincide very closely. A chi-square goodness of fit test on the observed distribution shows great significance. This example has been used in many statistics texts over the years. Its relevance has diminished in a practical sense (there aren't as many horse corps as there once were), but its importance in the history of statistics, along with Poisson, is well assured. Incidentally, Poisson died in 1840 near Paris.

But do the sales data really follow a Poisson distribution? To find out, we first need to know the density function (usually called "mass" function for discrete distributions) for the Poisson. It is

$$\frac{e^{-\mu}\mu^{x}}{x!}$$

where μ is the mean of the distribution (usually estimated as $\bar{x}$ from the sample data), x is the count of the number of orders per day ($x = 1, 2, 3, \ldots$), $e^{-\mu}$ is the exponential constant (= 2.71828) raised to the power -μ, and $x!$ is "x factorial" (defined as $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdots x$, with 0! defined as equal to 1). Although the expression may look horrible, it's easy to calculate on a hand calculator. First, though, we need to know the mean of the 75 data values (the average number of orders per day). We can add all 75 values and divide by 75, or take advantage of the work we already expended to develop the histogram. Thus

$$\bar{x} = \frac{4 \cdot 0 + 4 \cdot 1 + 6 \cdot 2 + 14 \cdot 3 + 18 \cdot 4 + 10 \cdot 5 + 11 \cdot 6 + 3 \cdot 7 + 1 \cdot 8 + 2 \cdot 9 + 1 \cdot 10 + 1 \cdot 12}{75}$$

$$= 4.2 \text{ orders per day}$$

$\bar{x}$ is an estimate of μ. Although it's not a nice whole number, that's O.K. It's usually quoted as a decimal fraction anyway. (Incidentally, the standard deviation of the 75 values is 2.27 orders per day.) As a demonstration, let's calculate the Poisson expression for $x = 6$ orders per day, as follows:

$$\frac{e^{-4.2}(4.2)^6}{6!} = \frac{(0.0150)(5,489.0317)}{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6} = 0.1143$$

If this number is multiplied by 75, we obtain the number of days out of 75 we could expect 6 orders, if the distribution of orders per day is exactly Poisson with mean 4.2 orders per day. The result is about 8.5741, or nearly 9 days. If we repeat this procedure for $x = 0, 1, 2$, etc., multiplying each result by 75 as above, we generate Table 3-12. Now we have *expected* occurrences to compare to those actually *observed* in the sample. Notice that since no days were recorded in the sample when more than 12 orders were placed, the observed value is 0. However, the theoretical Poisson is an infinite distribution, so it requires a small estimate even beyond 12. This is equal to 75 less the sum of all the expected values from 0 to 12, or about 0.0323.

Table 3-12. Observed and Expected Daily Sales Assuming a Poisson Model

NO. OF ORDERS PER DAY (*X*)	NO. OF DAYS OBSERVED WITH *X* ORDERS	POISSON EXPRESSION FOR *X*	NO. OF DAYS EXPECTED WITH *X* ORDERS	CUMULATIVE POISSON VALUES FROM COLUMN 3
0	4	0.01500	1.1247	0.01500
1	4	0.06298	4.7236	0.07798
2	6	0.13226	9.9196	0.21024
3	14	0.18517	13.8874	0.39541
4	18	0.19442	14.5818	0.58983
5	10	0.16332	12.2487	0.75315
6	11	0.11432	8.5741	0.86747
7	3	0.06859	5.1444	0.93606
8	1	0.03601	2.7008	0.97207
9	2	0.01681	1.2604	0.98888
10	1	0.00706	0.5294	0.99594
11	0	0.00269	0.2021	0.99863
12	1	0.00094	0.0707	0.99957
> 12	0	0.00043	0.0323	1.00000

It is again recommended that contiguous expected values be combined to assure that each expected number of days with *x* orders is at least 5. The observed days are combined as well. This results in 7 comparisons of observed and expected values, as shown in Table 3-13. In this case, only the mean is needed to calculate the Poisson expression, so the number of degrees of freedom is equal to $7 - 1 - 1 = 5$. Finally, chi-square is calculated as before, from the combined intervals in Table 3-13:

$$\chi^2 = \frac{(8-5.85)^2}{5.85} + \frac{(6-9.92)^2}{9.92} + \mathrm{K} + \frac{(8-9.94)^2}{9.94}$$

$$= 4.621$$

For a 5% significance level and 5 degrees of freedom, the tabulated $\chi^2 = 11.070$. Since the calculated value is less than the table value, the distribution of orders per day is judged to be Poisson.

Table 3-13. Table 3-12 After Combining Intervals

NO. OF ORDERS PER DAY (*X*)	NO. OF DAYS OBSERVED WITH *X* ORDERS	POISSON EXPRESSION FOR *X*	NO. OF DAYS EXPECTED WITH *X* ORDERS
<1	8	0.07798	5.8485
2	6	0.13226	9.9196
3	14	0.18517	13.8874
4	18	0.19442	14.5818
5	10	0.16332	12.2487
6	11	0.11432	8.5741
>7	8	0.13253	9.9401

With this information, we might want to ask the following questions:

- How many units of the item should be kept in daily inventory to meet 95% of the demand?

To answer this question, we can accumulate the Poisson values in the third column of Table 3-12 until we reach a total of 0.95. The cumulative Poisson values are given in the last column of Table 3-12. This occurs between 7 and 8 orders per day, so holding 8 items in daily inventory would satisfy at least 95% of the demand. We can also obtain the answer to this question by executing Query 3_16, which is shown on the next page.

SQL/Query

Query 3_16:

```
SELECT Min([Table 3_12].[Orders Per Day]) AS [Daily Inventory]
FROM [Table 3_12]
WHERE [Table 3_12].[Cumm Poisson] >= 0.95;
```

- What percent of daily sales fall between 2 and 6 units per day?

A similar approach may be used to answer this question. Add the Poisson values from 2 through 6 to yield 0.78949. Thus about 79% of daily sales fall between 2 and 6 units. This can also be obtained by executing Query 3_17.

SQL/Query

Query 3_17:

```
SELECT Sum([Table 3_12].[Poisson])*100 AS [Percent of Daily Sales]
FROM [Table 3_12]

WHERE [Table 3_12].[Orders Per Day] Between 2 And 6;
```

- Is there a justification for keeping more than 15 units per day in inventory?

To answer this question, we need to calculate the Poisson expression for $x = 13$, 14, and 15, which we don't have yet. These will be very small numbers, as shown below:

$$\frac{e^{-4.2}(4.2)^{13}}{13!} = 0.0003047$$

$$\frac{e^{-4.2}(4.2)^{14}}{14!} = 0.0000914$$

$$\frac{e^{-4.2}(4.2)^{15}}{15!} = 0.0000256$$

From Table 3-12, we know the chance that the daily orders are less than or equal to 12 is the sum of the Poisson expression from 0 through 12, or 0.99957. Adding the three values above brings this total to 0.99999, so the likelihood of orders of more than 15 units in a day is 1-0.99999 = 0.00001, or 0.001% (a very slim chance). We can obtain this result by running Query 3_18. The Poisson function is defined in Appendix D.

SQL/Query

Query 3_18:

```
SELECT
Round(1 - ([Cumm Poisson] + Poisson(13,4.2) +
Poisson(14,4.2) + Poisson(15,4.2)), 5) AS [More than 15 units]
FROM [Table 3_12]
WHERE [Table 3_12].[ Orders Per Day] = 12;
```

Fitting an Exponential Distribution to Observed Data

A service company maintains an Internet Web site to attract potential customers. The webmaster is presently tracking the elapsed time (in minutes) between successive "hits" on the site. Fifty observations during the day generated the data in Table 3-14.

Table 3-14. Fifty Random Observations of Elapsed Times (in Minutes) Between "Hits" on an Internet Web Site

17	20	10	9	23	13	12	19	18	24
12	14	6	9	13	6	7	10	13	7
16	18	8	13	3	32	9	7	10	11
13	7	18	7	10	4	27	19	16	8
7	10	5	14	15	10	9	6	7	15

A histogram may be set up for these observations, as shown in Figure 3-7. We've used a convenient interval width of 5 minutes, again with the left-hand endpoint inclusive and the right-hand endpoint exclusive for each interval.

We note from the histogram that there were two occurrences where the time between successive hits was less than 5 minutes. The most representative elapsed time, however, appears to be between 5 and 20 minutes. The mean is calculated by summing the 50 observations and dividing the result by 50. This yields $\bar{x} = 12.32$ minutes as the average time between "hits." Since the data are continuous (variables data), an appropriate model should be selected from among the continuous models available. At first glance, the data values seem to be exponentially decreasing. Consequently, an exponential distribution is chosen initially to characterize the data. The density expression

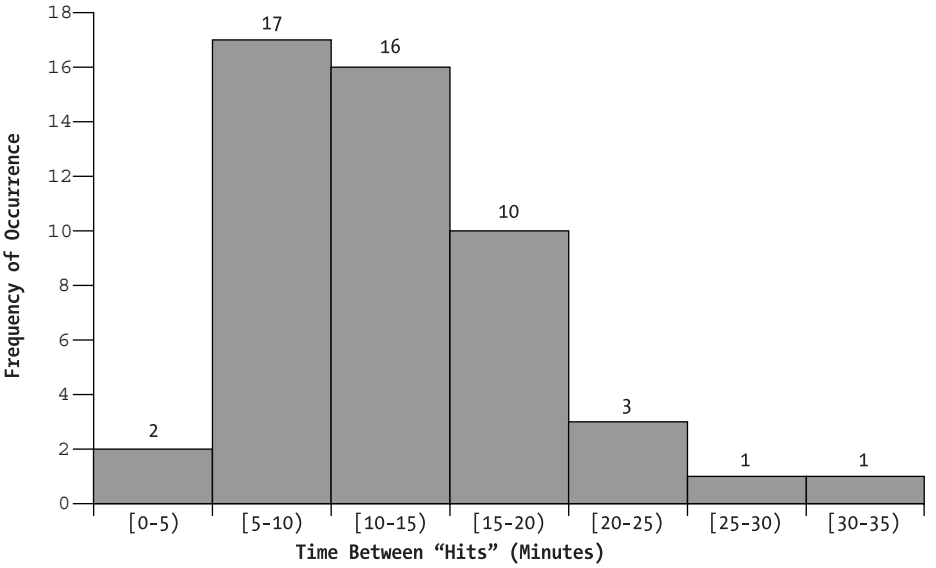

Figure 3-7. Histogram of times between hits from Table 3-14

for such a model is defined below in terms of the estimated mean $\overline{x}$ and the random variable x:

$$\frac{1}{\overline{x}} e^{-x/\overline{x}}, \quad x \geq 0$$

However, this is not the most convenient expression to yield the expected frequencies. In practice, we are required to integrate the above expression and multiply by 50 (the number of observations) to obtain the expected frequency in any particular interval of the histogram. Instead of that cumbersome approach, we can use the following expression instead:

$$50\left(1 - e^{-x/\overline{x}}\right)$$

This yields the cumulative expected frequency at any point (x value) on the histogram. The expression in parentheses is obtained by integrating

$$\frac{1}{\overline{x}} e^{-x/\overline{x}}$$

from 0 to x. It is known as the cumulative exponential distribution function.

To use it, we begin with the first interval from 0 to 5. Substituting $x = 5$ and $\bar{x} = 12.32$ into the cumulative distribution function, we obtain

$$50\left(1 - e^{-5/12.32}\right) = 16.68$$

This is the expected number of observations between 0 and 5 minutes. Next we calculate the expected number of observations between 0 and 10 minutes, as shown below:

$$50\left(1 - e^{-10/12.32}\right) = 27.79$$

By subtracting the expected number between 0 and 5 minutes from the expected number between 0 and 10 minutes, we obtain the expected number between 5 and 10 minutes, as follows:

$$27.79 - 16.68 = 11.11$$

Proceeding in this manner, we can generate Table 3-15.

Table 3-15. Observed and expected frequencies of times between successive "hits"

HISTOGRAM INTERVAL	OBSERVED FREQUENCY	EXPECTED FREQUENCY
[0 – 5)	2	16.68
[5 – 10)	17	11.11
[10 – 15)	16	7.41
[15 – 20)	10	4.94
[20 – 25)	3	3.29
[25 – 30)	1	2.19
[30 – 35)	1	1.46
≥ 35	0	2.92

Combining expected frequencies to insure each interval has a total of at least 5, we obtain Table 3-16. It also shows the χ^2 terms for each resulting interval.

Table 3-16. Calculation of Terms for the χ^2 Statistic

HISTOGRAM INTERVAL	OBSERVED FREQUENCY (O)	EXPECTED FREQUENCY (E)	$\frac{(O-E)^2}{E}$
[0 – 5)	2	16.68	12.920
[5 – 10)	17	11.11	3.123
[10 – 15)	16	7.41	9.958
[15 – 25)	13	8.23	2.765
≥ 25	2	6.57	3.179

The χ^2 statistic is the sum of the last column in Table 3-16, or about 31.94. The number of degrees of freedom is equal to 5 intervals (from Table 3-16) less one parameter (the mean) less one, or 3. For a 5% significance level and 3 degrees of freedom, the table χ^2 value is 7.815. Since the calculated value exceeds the table value, we conclude that the distribution of times between successive "hits" is significantly different from the exponential distribution. (That's statistician jargon to simply say "it didn't fit.") We therefore might want to try a different model for better characterization. It turns out that a gamma distribution of the form

$$0.00162x^{3.10}e^{-0.33x}$$

is a reasonably good fit. This could have been foreseen from the original histogram, since the shape was not continually decreasing (a characteristic of the exponential distribution). Rather, it started low, rose to a maximum, then tapered down to the right.

Conclusion

There are many distributions, either discrete or continuous, that are candidates for sample data characterization. The nature of the data and shape of the histogram are clues that help guide the data miner toward a possible model. The χ^2 test is then helpful in comparing the observed data distribution to a theoretical model. In Appendix C we present a list of some of the more common distributions and their characteristics, in a table format for easier use. There are many models available. Please understand that in some cases the mechanics of calculating expected frequencies for certain models might be quite

complicated, unless tables or cumulative distribution expressions are available. In addition, some of the distributions presented (such as χ^2, F, and Student's *t*) are more commonly associated with hypothesis testing than with characterizing data sets. Their density function expressions are pretty intimidating, as you can see from the table in Appendix C.

John's Jewels
The Godfather of Statistical Distributions

I'll tell you something you probably won't believe. There are dozens of statistical distributions, both discrete and continuous. However, there is one umbrella distribution that can be used to derive any of these known distributions. It is called the H function distribution. By substituting appropriate values for its parameters, we obtain the binomial, the Poisson, the normal, the exponential, and in fact any other statistical distribution, discrete or continuous. The procedure for accomplishing this, however, is quite complex, so the H function is little known and rarely encountered. It's like the godfather of the statistical family of distributions.

T-SQL Source Code

There are three T-SQL source code listings for performing the chi-square goodness of fit test for the normal distribution described in this chapter. The first listing is called Make_Intervals. Make_Intervals takes a table of values, groups the values based on the range (interval width), and counts the number of values within each interval. The second listing is called Combine_Intervals. Combine_Intervals takes the result of Make_Intervals, finds those intervals with a count value less than 5, and combines these smaller intervals with respect to their adjacent intervals until all intervals have a count greater than or equal to 5. The last listing is called Compare_Observed_And_Expected. Compare_Observed_And_Expected takes the results of Combine_Intervals and performs the statistical calculations necessary to compare the observed values to the expected values. Following the procedure listings is a section illustrating the call statements to each procedure with parameter specifications.

Make_Intervals

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS OFF
GO

CREATE PROCEDURE Make_Intervals
@Interval_Size smallint = 7,
@SourceTblName  VarChar(50) = '[Limo Miles]',
@SourceColName  VarChar(50) = '[Miles]',
@ResultTblName  VarChar(50) = '[Table 3_1]'
AS

/*********************************************************/
/*                                                       */
/*                     MAKE INTERVALS                    */
/*                                                       */
/*   This procedure takes a table of values, groups the  */
/* values into intervals of a specified width, and then  */
/* counts the number of values within each interval.     */
/*                                                       */
/* INPUTS:                                               */
/*   SourceTblName - table containing sample data        */
/*   SourceColName - column containing sample data values*/
/*   ResultTblName - table to receive intervals          */
/*                                                       */
/* NOTE:                                                 */
/* The result table has four columns:                    */
/*   Interval - interval ID number begins with zero      */
/*   LowEnd - lower bound of interval                    */
/*   HiEnd - upper bound of interval                     */
/*   ResultColName - (see INPUTS above)                  */
/*                                                       */
/*********************************************************/

/* Clear the result table */
EXEC('DELETE  FROM ' + @ResultTblName)
```

```
/* This query forms the intervals */
EXEC('INSERT INTO '+ @ResultTblName +
     'SELECT Floor((' + @SourceColName + ')/' +
               @Interval_Size + ') AS Interval, ' +
     '(Floor((' + @SourceColName + ')/' + @Interval_Size +
               '))*'+@Interval_Size + ' AS LowEnd, ' +
     '(Floor((' + @SourceColName + ')/' + @Interval_Size +
               '+1))*' + @Interval_Size + 'AS HiEnd, ' +
     'Count(' + @SourceTblName + '.' + @SourceColName +
               ') AS CountOf ' +
     'FROM ' + @SourceTblName + ' ' +
     'GROUP BY Floor((' + @SourceColName + ')/' +
               @Interval_Size + '), ' +
     '(Floor((' + @SourceColName + ')/' + @Interval_Size +
               '))*' + @Interval_Size + ', ' +
     '(Floor((' + @SourceColName + ')/' + @Interval_Size +
               '+1))*' + @Interval_Size + ' ' +
     'ORDER BY Floor((' + @SourceColName + ')/' +
               @Interval_Size + ')')

GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

Combine_Intervals

```
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

CREATE PROCEDURE Combine_Intervals
@Src1TblName  VarChar(50) = 'Limo Miles',
@Src1ColName  VarChar(50) = 'Miles',
@Src2TblName  VarChar(50) = 'Table 3_1',
@RstTblName  VarChar(50) = 'Table 3_3'
AS
```

```
/************************************************************/
/*                                                          */
/*                   COMBINE INTERVALS                      */
/*                                                          */
/*   This procedure takes the result table from the         */
/* Make_Interval procedure and combines those intervals     */
/* with an expected frequency less than 5 with their        */
/* adjacent intervals until all intervals have values       */
/* greater than or equal to 5.                              */
/*                                                          */
/* INPUTS:                                                  */
/*   Src1TblName - table name for sample data               */
/*   Src1ColName - column name for sample data              */
/*   Src2TblName - result table name from Make_Interval     */
/*   RstTblName - result table name from combined interval  */
/*                                                          */
/************************************************************/

/* Local Variables */
DECLARE @Q Varchar(500)          /* Query string */
DECLARE @Xbar Float              /* Sample mean */
DECLARE @SD Float                /* Sample Standard deviation */
DECLARE @N Int                   /* Number of sample data values */
DECLARE @ID Int                  /* Interval counter */
DECLARE @LO Int                  /* Lower bound of interval being combined */
DECLARE @HI Int                  /* Upper bound of interval being combined */
DECLARE @Lowest Int              /* Lower bound of all intervals */
DECLARE @Highest Int             /* Upper bound of all intervals */
DECLARE @SumO Int                /* Sum of the sum of observed vallues */
DECLARE @SumE Float              /* Sum of the expected frequencies */
DECLARE @Sum_Less_Last Float     /* Sum E's except last one */

/* SETUP FOR INTERVALS */

/* Obtain sample mean and standard deviation */
SET @Q = 'SELECT Avg(Convert(Float, [' + @Src1ColName + ''])) AS Xbar, ' +
     'StDev([' + @Src1ColName + ']) AS SD, ' +
     'Count([' + @Src1ColName + ']) AS N ' +
     'INTO ##TempXbar_SD_N ' +
     'FROM [' + @Src1TblName + '] '
EXEC(@Q)
SELECT @Xbar = Xbar, @SD = SD, @N = N FROM ##TempXbar_SD_N
```

```
/* Establish intermediate work table */
CREATE TABLE ##TempWork (Interval Int,
     LowEnd Int, HiEnd Int, CountOf Int,
     Xbar Float, SD Float, NVariate Float, StdNorm Float null)

/* Populate the work table */
SET @Q = 'INSERT INTO ##TempWork (Interval, LowEnd, HiEnd, ' +
     'CountOf, Xbar, SD, NVariate, StdNorm) ' +
     'SELECT Interval, LowEnd, HiEnd, CountOf, ' +
     str(@Xbar,8,3) + ' AS Xbar, ' +
     str(@SD,8,3) + 'AS SD, ' +
     '(HiEnd - ' + str(@Xbar,8,3) + ')/' +
     str(@SD,8,3) + ' AS NVariate, 0.0 AS StdNorm ' +
     'FROM [' + @Src2TblName + ']'
EXEC(@Q)

/* Update work table with Standard Normal */
UPDATE ##TempWork
SET StdNorm =
     (Select Area FROM StdNormal
     WHERE Str(Nvariate,9,3) = Str(StdNormal.X,9,3))

/* Get lowest, highest interval values */
SELECT @Lowest = Min([Interval]),
     @Highest = Max([Interval])
     FROM ##TempWork

/* Establish table of observed (O) and */
/* expected frequencies (E) and insert */
/* the first interval */
SET @Q = 'SELECT ##TempWork.Interval As Interval, ' +
     '##TempWork.LowEnd AS LowEnd, ' +
     '##TempWork.HiEnd AS HiEnd, ' +
     '##TempWork.CountOf AS O, ' +
     '##TempWork.StdNorm * ' + str(@N,8,1) + ' AS E ' +
     'INTO ##TempTable3_3 ' +
     'FROM ##TempWork ' +
     'WHERE ##TempWork.Interval = ' + Convert(Varchar(20), @Lowest)
EXEC(@Q)
```

```
/* Calculate the expected frequency for */
/* all of the in-between intervals */
SET @Q = 'INSERT INTO ##TempTable3_3 (Interval, LowEnd, HiEnd, O, E) ' +
     'SELECT ##TempWork.Interval AS Interval, ' +
     '##TempWork.LowEnd AS LowEnd, ' +
     '##TempWork.HiEnd AS HiEnd, ' +
     '##TempWork.CountOf as O, ' +
     '(##TempWork.StdNorm - ##TempWork_1.StdNorm) * ' +
     str(@N,8,1) + ' AS E ' +
     'FROM ##TempWork, ##TempWork ##TempWork_1 ' +
     'WHERE ##TempWork.Interval = (##TempWork_1.Interval + 1) ' +
     'AND ##TempWork.Interval > ' + Convert(Varchar(20), @Lowest) + ' ' +
     'AND ##TempWork.Interval < ' + Convert(Varchar(20), @Highest)
EXEC(@Q)

/* Sum all the preceding expected frequences (E) */
/* before appending the last interval */
SELECT @Sum_Less_Last = Sum(E) FROM ##TempTable3_3

/* Determine expected frequency for last interval */
/* and then append the last interval */
SET @Q = 'INSERT INTO ##TempTable3_3 (Interval, LowEnd, HiEnd, O, E) ' +
     'SELECT ##TempWork.Interval AS Interval, ' +
     '##TempWork.LowEnd AS LowEnd, ' +
     '999 AS HiEnd, ' +
     '##TempWork.CountOf as O, ' +
     Convert(Varchar(20), @N) + ' - ' +
     str(@Sum_Less_Last,8,3) + ' AS E ' +
     'FROM ##TempWork ' +
     'WHERE ##TempWork.Interval = ' + Convert(Varchar(20), @Highest)
EXEC(@Q)

/* COMBINE INTERVALS */

/* If result table exists, then drop it */
IF exists (SELECT id FROM ..sysobjects
          WHERE name = @RstTblName)
Begin
     SET @Q = 'DROP TABLE [' + @RstTblName + ']'
     EXEC(@Q)
End
```

```
/* Create the result table */
SET @Q = 'CREATE TABLE [' + @RstTblName + '] (Interval Int, '+
     'LowEnd Int, HiEnd Int, O Float, E Float)'
EXEC(@Q)

/* Define the cursor and the local     */
/* variables to receive the row values */
DECLARE Cr1 INSENSITIVE SCROLL CURSOR
     FOR SELECT Interval, LowEnd, HiEnd, O, E
     FROM ##TempTable3_3
     ORDER BY Interval
DECLARE @Intv1 Int, @Low1 Int, @Hi1 Int, @O1 Float, @E1 Float

OPEN Cr1

/* Set cursor to first record */
FETCH NEXT FROM  Cr1
     INTO @Intv1, @Low1, @Hi1, @O1, @E1

/* Initialize */
SET @ID = 0

/* Go through the table and combine those  */
/* intervals that have a count less than 5 */

WHILE @@FETCH_Status = 0
     Begin
          If @E1 >= 5
          Begin
               /* Do not combine, just copy over  */
               /* the interval and move cursor    */
               /* ahead one row and get next row. */
               SET @Q = 'INSERT INTO [' + @RstTblName + '] ' +
                    'VALUES(' + str(@ID,8,0) + ', ' +
                    str(@Low1,8,0) + ', ' +
                    str(@Hi1,8,0) + ', ' +
                    str(@O1,12,5) + ', ' +
                    str(@E1,12,5) + ')'
               EXEC(@Q)

               SELECT @ID = @ID + 1

               FETCH NEXT FROM Cr1
                    INTO @Intv1, @Low1, @Hi1, @O1, @E1
          End
```

```
Else
Begin
     /* Combine the two or more intervals into   */
     /* one interval until we have @SumE >= 5,   */
     /* also remember low point of this interval */
     SELECT @LO = @Low1
     SELECT @SumO = 0.0
     SELECT @SumE = 0.0

     While (@@Fetch_Status = 0 and @SumE < 5)
     Begin
          SELECT @SumO = @SumO + @O1
          SELECT @SumE = @SumE + @E1
          SELECT @HI = @Hi1

          FETCH NEXT FROM Cr1
               INTO @Intv1, @Low1, @Hi1, @O1, @E1
     End

     /* Did we exit the loop */
     /* because End-Of-Table? */
     If @@Fetch_Status  <> 0 and @SumE < 5
     Begin
          /* Combine with the last row since */
          /* there are no more intervals to  */
          /* combine to get a value >= 5     */
          SET @Q = 'UPDATE [' + @RstTblName + '] ' +
          'SET O = O + ' + str(@SumO,12,5) + ', ' +
          'E = E + ' + str(@SumE,12,5) + ', ' +
          'HiEnd = ' + str(@HI,8,0) + ' ' +
          'WHERE HiEnd = ' + Convert(varchar(20), @LO)
          EXEC(@Q)
     End
```

```
                Else
                Begin
                     /* Save the combined interval */
                     SET @Q = 'INSERT INTO [' + @RstTblName + '] ' +
                     'VALUES(' + str(@ID,8,0) + ', ' +
                     str(@LO,8,0) + ', ' +
                     str(@HI,8,0) + ', ' +
                     str(@SumO,12,5) + ', ' +
                     str(@SumE,12,5) + ')'
                     EXEC(@Q)
                End

          End

End  /* End While loop */

Close Cr1
Deallocate Cr1

/* Replace CombInterval with the RemInterval */
Delete FROM [CombInterval]
INSERT INTO [CombInterval]
     SELECT * FROM RemInterval

GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON

GO
```

Compare_Observed_And_Expected

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

CREATE PROCEDURE [Compare_Observed_And_Expected]
@Alpha Float = 0.05,
@v Int = 7,
@Src1TblName VarChar(50) = 'Table 3_3'
AS
```

```
/*************************************************************/
/*                                                           */
/*        COMPARE OBSERVED AND EXPECTED FREQUENCIES          */
/*                    FOR A GOODNESS OF FIT                   */
/*               TO THE NORMAL DISTRIBUTION                  */
/*                                                           */
/*   This procedure takes the result table from the          */
/* Combine_Intervals procedure and determines whether or     */
/* not the data fits a normal distribution. This is          */
/* accomplished by performing a Chi-square test between the */
/* observed frequencies and the expected frequencies.        */
/*                                                           */
/* INPUTS:                                                   */
/*   Src1TblName - result table from Combine_Intervals       */
/*                                                           */
/*  TABLES:                                                  */
/*   TableChiSQ - Chi square values                          */
/*      Contents:                                            */
/*      CalcChiSq - float, calculated chi square             */
/*      TableChSq - float, table value of chi square         */
/*   ChiSquare -- the statistical table of Chi Squares.      */
/*      Contents:                                            */
/*   Alpha - significance level                              */
/*      v - degrees of freedom                               */
/*      ChiSq - chi square for Alpha and v                   */
/*                                                           */
/*************************************************************/

/* Local Variables */
DECLARE @CalcChiSq float     /* Calculate Chi Square */
DECLARE @TableChiSq float     /* Table Chi Square value */

/* Calculate Chi Square */
EXEC('SELECT Sum(Power(([O]-[E]),2)/[E]) AS V ' +
     'INTO ##TmpChiSQCalcTable ' +
     'FROM [' + @Src1TblName + '] ')
SELECT @CalcChiSq = V
     FROM ##TmpChiSQCalcTable
```

```
/* Look up the Chi Square table value */
EXEC('SELECT ChiSquare.ChiSq AS TableChiSq ' +
     'INTO ##TmpChiSQTblTable ' +
     'FROM ChiSquare ' +
     'WHERE ((ChiSquare.[Percent]=' + @Alpha + ') ' +
     'AND (ChiSquare.Degress_Of_Freedom=' + @v + '))')
SELECT @TableChiSq = TableChiSq
     FROM ##TmpChiSQTblTable

/* Save the Chi Square values, */
/* but first clear the table */
DELETE [ChiSq For Data]
INSERT INTO [ChiSq For Data]
     SELECT @CalcChiSq, @TableChiSq

GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

Procedure Calls

Below are the call statements for calling the procedures Make_Intervals, Combine_Intervals, and Compare_Observed_And_Expected.

```
DECLARE @RC int
DECLARE @Interval_Size int
DECLARE @SourceTblName varchar(50)
DECLARE @SourceColName varchar(50)
DECLARE @ResultTblName varchar(50)
EXEC @RC = [CH3].[dbo].[Make_Intervals] @Interval_Size=7,
@SourceTblName='[Limo Miles]', @SourceColName='[Miles]',
@ResultTblName='[Table 3_1]'
```

```
DECLARE @RC int
DECLARE @Src1TblName varchar(50)
DECLARE @Src1ColName varchar(50)
DECLARE @Src2TblName varchar(50)
DECLARE @RstTblName varchar(50)
EXEC @RC = [CH3].[dbo].[Combine_Intervals] @Src1TblName ='Limo Miles',
@Src1ColName='Miles',
@Src2TblName ='Table 3_1', @RstTblName='Table 3_3'

DECLARE @RC int
DECLARE @Alpha float
DECLARE @v int
DECLARE @Src1TblName varchar(50)
EXEC @RC = [CH3].[dbo].[Compare_Observed_And_Expected]
@Alpha=0.05, @v=7, @Src1TblName='Table 3_3'
```