```
'LCAG activity report - John E. Carter
'email: johnecarter@mindspring.com  pager: 770-589-9817
'If the processing of an item covers more than one day, the total time is
'the difference between start time and COB on the first day, and OOB and end time
'on the last day.  Any additional work days between first and last days are
'counted at the number of hours between OOB and COB.
'This is implemented in Release 2 - times on start and end days are
'computed to the minute, intervening days (except Saturday and Sunday)
'are counted as 12 hours.
'Holidays are NOT recognized.  This requires a lookup table that must be
'updated annually.
'
'The production database is CLDS_PROD_DB on chaucer.
'11-19-97 - jec - initial release
'12-01-97 - jec - added last request time to Status sheet
'12-02-97 - jec - added "Today" button to select all records from midnight to current time
'12-02-97 - jec - added check for start and end date/time being greater than current date/time
'12-03-97 - jec - added check for current minute being 00 in get_today()
'12-04-97 - jec - added reset button to set dropdowns to first day of current month
'              - deleted unneeded add-ins
'12-11-97 - jec - added select of cell a6 on Data sheet to ensure first row of data
'                is displayed on each new request
'12-16-97 - jec - SQL query now saved in "{application.path}\sqlquery.txt"
'
'01-05-98 - jec - 1.6 - added Pending chart to application
'
'01-14-98 - jec - 1.7
'              - added clear of cell b1 on Data sheet (user ID) to clear_data()
'              - added force of user ID (Data.b2) to lowercase
'              - changed name in Data sheet header to "LCAG Report"
'
'02-05-98 - jec - 1.8 - added Counts function
'
'02-10-98 - jec - 1.9 - deleted Lcase() of user_id$
'
'02-12-98 - jec - 1.10 - added "days present" option in getcount()
'
'02-27-98 - jec - 1.10a - minor cleanups in comments
'
'04-27-98 - jec - 1.11 - added "In Progress Time"
'
'04-28-98 - jec - 1.11a - changed color coding in "In Progress Time" cells to
'                         magenta for >= 1 minute and red for >= 60 minutes
'
'05-04-98 - jec - 1.11b - changed title of "Overview Chart" to "Overview"
'
'05-05-98 - jec - 1.12 - added check for current year in validating month/day
'                      - added cells in "Counts" sheet to clear_data sub
'                      - extended "Pending" and "In Progress" charts to 500 rows
'
'05-18-98 - jec - 1.12a - locked heading cells on "Data" page
'
'06-02-98 - jec - 1.13 - added option to get counts for all users over specified time
'                      - added progress indicator for counts function
'
'06-09-98 - jec - 1.13a - changed 'endrange' in copy_data to use count from Data!i1
'                         to prevent incomplete data copy when another function
'                         (counts, days) is called between data retrieval and start of
'                         sort request
'
'06-10-98 - jec - 1.14 - changed processing to use VBA code instead of macros to compute
'                        process time and do status-code-to-text conversion
```

```
'                   (on a Pentium 90 PC, the VBA code processes about 20 rows/second)
'                 - need to add checks for time of day, day of week, and holiday
'                   exclusions.
'
'06-16-98 - jec - 1.15 - added label to display progress messages
'
'06-17-98 - jec - 1.16 - added code to compute overnight times based on OOB and COB hours
'                   (overnight ONLY - does NOT handle multiple days between events)
'
'06-22-98 - jec - 1.17 - moved all time handling code into function compute_time()
'
'06-23-98 - jec - 1.18 - added leap year and month change checks to compute_time
'
'08-03-98 - jec - 1.19 - production system moved from chaucer to as01 - changed host
'                   name in SQLOpen statements in Module1 and Module2
'
'08-04-98 - jec - 1.20 - set calc to automatic to update charts
'
'08-07-98 - jec - 1.21 - added code to compute_time() to handle requests received
'                   on current day but before OOB
'
'08-10-98 - jec - 1.22 - added counts of pending and in progress items
'
'08-12-98 - jec - 1.23 - added error message for server not available
'
'08-13-98 - jec - 1.24 - added improved error handling for server unavailable errors
'
'08-11-99 - jec - 2.00 - times include OOB-COB periods on weekdays only, weekend
'                   days are excluded

'common variables
Public isleap As Integer 'used in main module and in computing times between days

' get_data_Click Macro
'
Sub get_data_Click()
'  Application.Calculation = xlManual
'  Application.CalculateBeforeSave = False
  test01
End Sub

'
' test01 Macro
'
Sub test01()
'default start date/time is current date at 00:00:00
'default end date/time is current
'defaults are used if the "Today" button is pressed

1 On Error GoTo errorhandler
Dim my_hour As Integer, my_minute As Integer, my_month As Integer, my_day As Integer
Dim my_year As Integer, monthnum As Integer
Dim yearnum As Integer, maxday As Integer, isleap As Integer, req_hour As Integer
Dim req_minute As Integer, servername As String, MySQLopen As String
5 Dim wksht As Object
Set wksht = Worksheets("Data")
servername = "as01" 'need entries in c:\windows\hosts and in ODBC table
20 Sheets("Data").range("b1").Select 'unused cell
'label used for progress messages is not attached to the undelying cell
Sheets("Data").Labels("Label 41").Placement = xlFreeFloating
Sheets("Data").Labels("Label 41").Text = "Verifying data range"
Sheets("Data").Labels("Label 41").Visible = True
```

```
DoEvents  'allow hourglass cursor to show

'selected items are added here from the dialog boxes
'setup to check that requested date/time is not greater than current date/time
curdate$ = Date$
curtime$ = Time$
my_hour = Val(Left$(curtime$, 2))
my_minute = Val(Mid$(curtime$, 4, 2))
my_month = Val(Left$(curdate$, 2))
my_day = Val(Mid$(curdate$, 4, 2))
my_year = Val(Right$(curdate$, 4))


'Start Date/Time
'check for leap year
monthnum = Val(wksht.DropDowns("start_month").List(wksht.DropDowns("start_month").ListIndex))
yearnum = Val(wksht.DropDowns("start_year").List(wksht.DropDowns("start_year").ListIndex))
Select Case monthnum
Case 1, 3, 5, 7, 8, 10, 12
  maxday = 31
Case 4, 6, 9, 11
  maxday = 30
Case 2
  isleap = yearnum Mod 4
  If isleap = 0 Then            'standard leap year
    iscent = yearnum Mod 100    'but not on century year
    If iscent = 0 Then
      isleap = yearnum Mod 400  'unless divisible by 400
    End If
  End If
  If isleap = 0 Then
    maxday = 29
  Else
    maxday = 28
  End If
End Select

'Message box is used to notify user of invalid date/time entry.
'Error is handled by exiting sub.  User can then re-enter date/time and try again.
daynum = Val(wksht.DropDowns("start_day").List(wksht.DropDowns("start_day").ListIndex))
If daynum > maxday Then
  MsgBox "Invalid start day: " & Format$(daynum), 64, "CLDS Metrics Report"
  Exit Sub
End If

If (monthnum = my_month And daynum > my_day And yearnum = my_year) Or (yearnum = my_year And monthnum >
my_month) Or yearnum > my_year Then
  MsgBox "Requested start date is greater than today's date.", 64, "CLDS Metrics Report"
  End
End If

req_hour = Val(wksht.DropDowns("start_hour").List(wksht.DropDowns("start_hour").ListIndex))
req_minute = Val(wksht.DropDowns("start_minute").List(wksht.DropDowns("start_minute").ListIndex))
If DateSerial(my_year, my_month, my_day) = DateSerial(yearnum, monthnum, daynum) Then
  If (req_hour > my_hour) Or (req_hour = my_hour And req_minute > my_minute) Then
    MsgBox "Requested start time is later than current time.", 64, "CLDS Metrics Report"
    End
  End If
End If

'build date/time string from content of drop downs
startime$ = wksht.DropDowns("start_month").List(wksht.DropDowns("start_month").ListIndex)
```

```
startime$ = startime$ & "/" & wksht.DropDowns("start_day").List(wksht.DropDowns("start_day").ListIndex)
startime$ = startime$ & "/" & wksht.DropDowns("start_year").List(wksht.DropDowns("start_year").ListIndex)
startime$ = startime$ & " " & wksht.DropDowns("start_hour").List(wksht.DropDowns("start_hour").ListIndex)
startime$ = startime$ & ":" & wksht.DropDowns("start_minute").List(wksht.DropDowns("start_minute").ListIndex)
startime$ = startime$ & ":00"

'End Date/Time - perform same checks as for start date/time
'check for leap year
monthnum = Val(wksht.DropDowns("end_month").List(wksht.DropDowns("end_month").ListIndex))
yearnum = Val(wksht.DropDowns("end_year").List(wksht.DropDowns("end_year").ListIndex))
Select Case monthnum
Case 1, 3, 5, 7, 8, 10, 12
  maxday = 31
Case 4, 6, 9, 11
  maxday = 30
Case 2
  isleap = yearnum Mod 4
  If isleap = 0 Then            'standard leap year
    iscent = yearnum Mod 100    'but not on century year
    If iscent = 0 Then
      isleap = yearnum Mod 400  'unless divisible by 400
    End If
  End If
  If isleap = 0 Then
    maxday = 29
  Else
    maxday = 28
  End If
End Select

daynum = Val(wksht.DropDowns("end_day").List(wksht.DropDowns("end_day").ListIndex))
If daynum > maxday Then
  MsgBox "Invalid end day: " & Format$(daynum), 64, "CLDS Metrics Report"
  Exit Sub
End If

If (monthnum = my_month And daynum > my_day And yearnum = my_year) Or (yearnum = my_year And monthnum >
my_month) Or yearnum > my_year Then
  MsgBox "Requested end date is greater than today's date.", 64, "CLDS Metrics Report"
  End
End If

req_hour = Val(wksht.DropDowns("end_hour").List(wksht.DropDowns("end_hour").ListIndex))
req_minute = Val(wksht.DropDowns("end_minute").List(wksht.DropDowns("end_minute").ListIndex))
If DateSerial(my_year, my_month, my_day) = DateSerial(yearnum, monthnum, daynum) Then
  If (req_hour > my_hour) Or (req_hour = my_hour And req_minute > my_minute) Then
    MsgBox "Requested end time is later than current time.", 64, "CLDS Metrics Report"
    End
  End If
End If

'build end date/time string from drop downs
endtime$ = wksht.DropDowns("end_month").List(wksht.DropDowns("end_month").ListIndex)
endtime$ = endtime$ & "/" & wksht.DropDowns("end_day").List(wksht.DropDowns("end_day").ListIndex)
endtime$ = endtime$ & "/" & wksht.DropDowns("end_year").List(wksht.DropDowns("end_year").ListIndex)
endtime$ = endtime$ & " " & wksht.DropDowns("end_hour").List(wksht.DropDowns("end_hour").ListIndex)
endtime$ = endtime$ & ":" & wksht.DropDowns("end_minute").List(wksht.DropDowns("end_minute").ListIndex)
endtime$ = endtime$ & ":00"

'build SQL query from drop downs and user id (if any)
```

```
mysql$ = "SELECT loc_cd_req_emp_wrk_asgn.lcr_ref_num, loc_cd_req_emp_wrk_asgn.emp_id,
loc_cd_req_emp_wrk_asgn.emp_wrk_asgn_dt_tm, loc_cd_req_emp_wrk_asgn.wrk_asgn_stat, loc_cd_req.lcr_stat_cd,
loc_cd_req.lcr_svc_typ"
'CLDS_USER_DB is database on the test server (dante)
'mysql$ = mysql$ & " FROM CLDS_USER_DB.dbo.loc_cd_req loc_cd_req,
CLDS_USER_DB.dbo.loc_cd_req_emp_wrk_asgn loc_cd_req_emp_wrk_asgn"
mysql$ = mysql$ & " FROM CLDS_PROD_DB.dbo.loc_cd_req loc_cd_req,
CLDS_PROD_DB.dbo.loc_cd_req_emp_wrk_asgn loc_cd_req_emp_wrk_asgn"
mysql$ = mysql$ & " WHERE emp_wrk_asgn_dt_tm >= '" & startime$ & "'"
mysql$ = mysql$ & " AND emp_wrk_asgn_dt_tm <= '" & endtime$ & "'"
mysql$ = mysql$ & " AND loc_cd_req.lcr_ref_num = loc_cd_req_emp_wrk_asgn.lcr_ref_num"

'add user ID if any
'user_id$ = ""
user_id$ = Sheets("Data").range("b2")
If user_id$ <> "all" And user_id$ <> "" Then
  mysql$ = mysql$ & " AND emp_id = '" & Trim(user_id$) & "'"
  Sheets("Data").range("b2") = user_id$
End If
mysql$ = mysql$ & " ORDER BY  lcr_ref_num, emp_wrk_asgn_dt_tm"

'check for valid date/time ranges - end after start, end no later than current, etc.
If DateValue(startime$) > DateValue(endtime$) Then
  MsgBox "Start date greater than end date: " & Left$(startime$, 10) & " > " & Left$(endtime$, 10), 64, "CLDS Metrics
Report"
  Exit Sub
End If
If (DateValue(startime$) = DateValue(endtime$)) And (TimeValue(startime$) >= TimeValue(endtime$)) Then
  MsgBox "Start time greater than or equal to end time: " & Mid$(startime$, 12, 5) & " >= " & Mid$(endtime$, 12, 5), 64,
"CLDS Metrics Report"
  Exit Sub
End If

'save current date/time
Sheets("Status").range("h1") = Now()
Sheets("Status").range("h2").ClearContents

'clear area for incoming data
If Sheets("Data").range("l1") > 0 Then 'Data!l1 has number of rows in most recent query, if any
  clearange$ = "a6:j" & Trim(Str$(Sheets("Data").range("l1") + 5))
Else
  clearange$ = "a6:j2500"  'not likely to have more than this
End If
Sheets("Data").range("i1").ClearContents
90 Sheets("Data").range(clearange$).ClearContents

Sheets("Data").Labels("Label 41").Text = "Opening connection"
100 DoEvents

'open connection - valid connection is a small value, usually less than 10
'dante is server for test database
'110 thisconn = SQLOpen("DSN=CLDSUTEST;UID=cldstestid;SRVR=dante;DB=CLDS_USER_DB;PWD=johnray1", , 2)
'115 thisconn = SQLOpen("DSN=CLDSPROD;UID=cldsrep;SRVR=chaucer;DB=CLDS_PROD_DB;PWD=cldsrep", , 2)
'changed SRVR to as01 on 8/3/98
MySQLopen = "DSN=CLDSPROD;UID=cldsrep;SRVR=" & servername & ";DB=CLDS_PROD_DB;PWD=cldsrep"
115 thisconn = SQLOpen(MySQLopen, , 2)
If IsError(thisconn) Then Error 9999  'invalid response from server
120 Sheets("Status").range("b2") = thisconn

'error messages from each SQL operation are placed on Status sheet
130 Sheets("Status").range("b15") = sqlerror()
```

```
'Sheets("Data").range("f6") = "Executing query"
Sheets("Data").Labels("Label 41").Text = "Executing query"

'send query
140 Sheets("Status").range("b4") = sqlexecquery(thisconn, StringToArray(mysql$))
145 Sheets("Status").range("b17") = sqlerror()
'bind columns to fields in the rows to be returned
150 Sheets("Status").range("b5") = sqlbind(thisconn, 1, Sheets("Data").range("a6"))
Sheets("Status").range("b19") = sqlerror()
160 Sheets("Status").range("c5") = sqlbind(thisconn, 2, Sheets("Data").range("b6"))
Sheets("Status").range("b21") = sqlerror()
170 Sheets("Status").range("d5") = sqlbind(thisconn, 3, Sheets("Data").range("c6"))
Sheets("Status").range("b23") = sqlerror()
180 Sheets("Status").range("e5") = sqlbind(thisconn, 4, Sheets("Data").range("d6"))

Sheets("Status").range("b25") = sqlerror()
190 Sheets("Status").range("f5") = sqlbind(thisconn, 5, Sheets("Data").range("e6"))
Sheets("Status").range("b27") = sqlerror()
200 Sheets("Status").range("g5") = sqlbind(thisconn, 6, Sheets("Data").range("f6"))
Sheets("Status").range("b29") = sqlerror()

'retrieve data into cells
210 Sheets("Status").range("b6") = sqlretrieve(thisconn, Sheets("Data").range("a6"), , , False, , , False)
Sheets("Status").range("b31") = sqlerror()
215 If Not IsEmpty(Sheets("Status").range("b6")) Then 'save the count of returned rows
      Sheets("Data").range("l1") = Sheets("Status").range("b6")
      Sheets("Data").range("i1") = Sheets("Status").range("b6")
End If

Sheets("Data").Labels("Label 41").Text = "Closing connection"
DoEvents
'close connection
220 Sheets("Status").range("b7") = sqlclose(thisconn)
Sheets("Status").range("b33") = sqlerror()

'ensure first row is displayed
Sheets("Data").range("a6").Select
Sheets("Data").range("b1").Select
'calculate values - runs faster with auto-calc turned off
230 If Sheets("Status").range("b6") > 0 Then
      Sheets("Data").Labels("Label 41").Text = "Calculating values"
      net_work ("Data")
      Sheets("Data").Labels("Label 41").Text = ""
235 Else
     MsgBox "There are no records which match your query.", 64, "CLDS Metrics Report"
     Sheets("Data").range("b1").Select
     Exit Sub
240 End If

Sheets("Data").range("b1").Select
'save date/time of end of request
Sheets("Status").range("h2") = Now()

245 Exit Sub
errorhandler:
'If Err = 1000 And Erl = 215 Then
If Err = 9999 Then
  MsgBox "The server (" & servername & ") is not responding." & Chr$(10) & Chr$(13) & "Please try again later.", 16,
"CLDS Metrics Report"
  Sheets("Data").Labels("Label 41").Text = ""
  reset_count
Else
```

```
250    MsgBox "Encountered error " & Err & " at line " & Format$(Erl), 16, "CLDS Metrics Report"
End If
260  Err = 0
270  sqlclose (thisconn)
Sheets("Status").range("b33") = sqlerror()
End Sub


Sub net_data()
  net_work ("Data")
End Sub

Sub net_sort()
  net_work ("Sorted Data")
End Sub

'sub to perform equivalent of NETWORKDAYS() function
'work_sheet is name of sheet to process
'match_column is letter of column to use for matching reference number data
'source_column is letter of column that is the source of date/time fields for processing
'start_row is number of first row of data to process
'end_row is number of last row of data to process
'work_column is letter of column to receive computed time data
'This could be redefined with the following line and called with all the arguments if you do
'not want to embed the row/column data.
'Sub net_work(work_sheet, match_column, source_column, start_row, end_row, work_column)
Sub net_work(in_sheet)
  Dim ref_number As String, next_ref As String, net_range As String
  Dim work_sheet As String, match_column As String, source_column As String
  Dim fullstat_column As String, work_column As String
  Dim pending_column As String, progress_column As String, status_column As String, ref_status As String
  Dim start_row As Integer, end_row As Integer, st_day As Integer, st_month As Integer
  Dim st_hour As Integer, st_min As Integer, en_day As Integer, en_month As Integer
  Dim en_hour As Integer, en_min As Integer, oob_min As Integer, cob_min As Integer
  Dim row_count As Integer, diff_minutes As Integer, tmp_time1 As Integer, tmp_time2 As Integer
  Dim today_min As Integer, yest_min As Integer, pend_count As Integer, ip_count As Integer
  Dim st_date As Date, en_date As Date
  oob_min = 6 * 60  '6:00AM
  cob_min = 18 * 60 '6:00PM
  pend_count = 0
  ip_count = 0
  If in_sheet <> "" Then
    work_sheet = in_sheet
  End If
  If work_sheet = "Data" Then
    start_row = 6
    end_row = Sheets("Data").range("I1") + 5
  Else
    start_row = 1
    end_row = Sheets("Data").range("I1")
  End If

  match_column = "A"
  source_column = "C"
  work_column = "G"
  pending_column = "I"
  progress_column = "J"
  status_column = "E"
  fullstat_column = "H"

  If start_row > end_row Then
    MsgBox "Starting row greater than ending row", vbOKOnly, "Error calling sub net_work"
```

```
    Exit Sub
  End If

  Sheets(work_sheet).Select
  'turn off screen updating while doing calculations to speed up processing
  Application.ScreenUpdating = False

'status flags: CG - completed, CR - rejected, PA - pending pickup, PI - in progress
  DoEvents
  For row_count = start_row To end_row
    ref_status = Sheets(work_sheet).range(status_column & Format$(row_count))
    If ref_status = "PA" Then   'pending pickup
      pend_count = pend_count + 1
      Sheets(work_sheet).range(fullstat_column & Format$(row_count)) = "Pending"
      st_date = Sheets(work_sheet).range(source_column & Format$(row_count))
      en_date = Now
      Sheets(work_sheet).range(pending_column & Format$(row_count)) = compute_time(st_date, en_date)
    End If
    If ref_status = "PI" Then   'in progress
      ip_count = ip_count + 1
      Sheets(work_sheet).range(fullstat_column & Format$(row_count)) = "In Progress"
      st_date = Sheets(work_sheet).range(source_column & Format$(row_count))
      en_date = Now
      Sheets(work_sheet).range(progress_column & Format$(row_count)) = compute_time(st_date, en_date)
    End If
    If ref_status = "CR" Then 'rejected
      Sheets(work_sheet).range(fullstat_column & Format$(row_count)) = "Rejected"
    End If
    If ref_status = "CG" Then 'completed OK
      Sheets(work_sheet).range(fullstat_column & Format$(row_count)) = "Complete"
    End If
    ref_number = Sheets(work_sheet).range(match_column & Format$(row_count))
    next_ref = Sheets(work_sheet).range(match_column & Format$(row_count + 1))
    If ref_number = next_ref Then
      st_date = Sheets(work_sheet).range(source_column & Format$(row_count))
      en_date = Sheets(work_sheet).range(source_column & Format$(row_count + 1))
      Sheets(work_sheet).range(work_column & Format$(row_count + 1)) = compute_time(st_date, en_date)
      Sheets(work_sheet).range(progress_column & Format$(row_count)) = ""
      Sheets(work_sheet).range(pending_column & Format$(row_count)) = ""
    End If
  Next row_count
  'added total pending and in progress counts 8-10-98
  If work_sheet = "Data" Then
    Sheets(work_sheet).range("i4") = pend_count
    Sheets(work_sheet).range("j4") = ip_count
  End If
  'turn on screen updating to display calculated values
  Application.ScreenUpdating = True
End Sub

'compute_time returns the number of *business* minutes between two date/time stamps
Function compute_time(st_date, en_date)
  Dim st_mon As Integer, st_day As Integer, st_hour As Integer, st_min As Integer
  Dim en_mon As Integer, en_day As Integer, en_hour As Integer, en_min As Integer
  Dim st_year As Integer, en_year As Integer, st_dow As Integer, en_dow As Integer
  Dim tmp_time1 As Integer, tmp_time2 As Integer, yest_min As Integer, today_min As Integer
  Dim bus_days As Integer, tmp_day As Integer, oob_min As Integer, cob_min As Integer
  oob_min = 6 * 60   'start at 06:00
  cob_min = 18 * 60  'end at 18:00

  st_year = Year(st_date)
  st_mon = Month(st_date)
```

```
st_day = Day(st_date)
st_dow = WeekDay(st_date)
st_hour = Hour(st_date)
st_min = Minute(st_date)
en_year = Year(en_date)
en_mon = Month(en_date)
en_day = Day(en_date)
en_dow = WeekDay(st_date)
en_hour = Hour(en_date)
en_min = Minute(en_date)
If st_mon = en_mon Then  'within same month
  bus_days = en_day - st_day
Else    'month to month transition
  Select Case st_mon
    Case 2
      If isleap = 0 Then
        last_day = 29
      Else
        last_day = 28
      End If
    Case 4, 6, 9, 11
      last_day = 30
    Case 1, 3, 5, 7, 8, 10, 12
      last_day = 31
  End Select
  bus_days = (last_day - st_day) + en_day  'total days
End If
'Adjust number of days to skip weekend days - added 8/99
If st_dow = 1 And en_dow = 2 Then 'Sunday to Monday transition
  bus_days = 0
  st_hour = 6
  st_min = 0
End If
If st_dow = 7 And en_dow = 2 Then 'Saturday to Monday transition
  bus_days = 0
  st_hour = 6
  st_min = 0
'End If
ElseIf st_dow = 6 And en_dow = 2 Then  'Friday to Monday transition
  bus_days = 1
'End If
Else
  If st_dow > en_dow And bus_days > 2 Then 'any other "over the weekend" transaction
    bus_days = bus_days - 2
  End If
End If
If bus_days = 0 Then   'times in same day
  'determine minutes from OOB to start and from OOB to end, then compute difference
  'added next 4 lines 8/7/98 to coorectly handle requests received before OOB
  If st_hour < 6 Then  'check for time of receipt, modify if needed
    st_hour = 6
    st_min = 0
  End If
  tmp_time1 = (st_hour * 60) + st_min
  tmp_time2 = (en_hour * 60) + en_min
  compute_time = tmp_time2 - tmp_time1
Else   'times not in same day
       'determine the hour and minute differences and include OOB/COB computations
  If bus_days = 1 Then  'previous day
    If st_hour < 6 Then  'check for time of receipt, modify if needed
      st_hour = 6
      st_min = 0
```

```
      End If
      If st_hour < 18 Then  'compute time until COB
        tmp_time1 = (st_hour * 60) + st_min
        yest_min = cob_min - tmp_time1
      Else
        yest_min = 0
      End If
      If en_hour >= 6 Then  'compute time from OOB
        tmp_time2 = (en_hour * 60) + en_min
        today_min = tmp_time2 - oob_min
      Else
        today_min = 0
      End If
      compute_time = yest_min + today_min
    Else  'more than one day between events
      'need to handle holidays (no time charged against LCAG)
      'need lookup table for holidays - preferably in CLDS

      'start with time from receipt to COB on st_date
      If st_hour < 6 Then  'check for time of receipt, modify if needed
        st_hour = 6
        st_min = 0
      End If
      If st_hour < 18 Then  'compute time until COB
        tmp_time1 = (st_hour * 60) + st_min
        yest_min = cob_min - tmp_time1
      Else
        yest_min = 0
      End If

      'check days between st_date and en_date, skip weekends
      'add 12 hours for each work day


      'get time from OOB to completion on en_date
      If en_hour >= 6 Then  'compute time from OOB
        tmp_time2 = (en_hour * 60) + en_min
        today_min = tmp_time2 - oob_min
      Else
        today_min = 0
      End If

      'add them all together for total time


      'for now, just use fudge factor of 12 hours
      compute_time = 720  '12 hours
    End If
  End If
End Function

'SQL query as a string can only be 255 characters.
'This converts it to an array, which can be much larger.
'Found on one of the non-Microsoft Web sites about Excel.
Function StringToArray(Query As String) As Variant
 Const StrLen = 127       'Set the maximum string length for
                          'each element in the array to return
                          'to 127 characters.
 Dim NumElems As Integer
 Dim Temp() As String
                  'Divide the length of the string Query by StrLen and add
                  '1 to determine how many elements the String array Temp
```

```
                'should contain and redimension the Temp array to contain
                'this number of elements.
 NumElems = (Len(Query) / StrLen) + 1
 ReDim Temp(1 To NumElems) As String
                'Build the Temp array by sequentially extracting 127
                'segments of the Query string into each element of the
                'Temp array.
'Log function added by John E. Carter
 logpath$ = Application.Path & "\sqlquery.txt"
 Open logpath$ For Output As 1
 Print #1, Date$, Time$, "cldsprod.xls"
 For i = 1 To NumElems
   Temp(i) = Mid(Query, ((i - 1) * StrLen) + 1, StrLen)
   Print #1, Temp(i)
 Next i
 Print #1, ""
 Close 1
                'Set the function StringToArray to the Temp array so it
                'can be returned to the calling procedure
 StringToArray = Temp
End Function

Sub clearcells()
  Sheets("Data").range("a6:j2500").ClearContents
End Sub

Sub copy_data()
  'ensure top row is visible
  Sheets("Data").range("a6").Select
  Sheets("Data").range("b1").Select
  Sheets("Sorted Data").range("a1:j2500").ClearContents
  endrange = Sheets("Data").range("i1")
  copyrange$ = "a6:j" & Trim(Str$(endrange + 5))
  destrange$ = "a1:j" & Trim(Str$(endrange))
  Sheets("Data").range(copyrange$).Copy (Sheets("Sorted Data").range(destrange$))
  Sheets("Sorted Data").Select
End Sub

Sub clear4()
  Sheets("Sorted Data").range("a1:j2500").ClearContents
End Sub

'
' cldsuser Macro
' sort by user
'
Sub cldsuser()
    copy_data
    Sheets("Sorted Data").Select
    Columns("A:J").Select
    Selection.Sort Key1:=range("B1"), Order1:=xlAscending, Key2:=range _
        ("A1"), Order2:=xlAscending, Key3:=range("C1"), Order3:= _
        xlAscending, Header:=xlGuess, OrderCustom:=1, MatchCase:=False _
        , Orientation:=xlTopToBottom
    range("A1").Select
End Sub

'
' cldstype Macro
' sort by service type
'
Sub cldstype()
```

```
    copy_data
    Sheets("Sorted Data").Select
    Columns("A:J").Select
    Selection.Sort Key1:=range("F1"), Order1:=xlAscending, Key2:=range _
        ("A1"), Order2:=xlAscending, Key3:=range("C1"), Order3:= _
        xlAscending, Header:=xlGuess, OrderCustom:=1, MatchCase:=False _
        , Orientation:=xlTopToBottom
    range("A1").Select
End Sub
'
' cldstime Macro
' sort by time and reference number
'
Sub cldstime()
    copy_data
    Sheets("Sorted Data").Select
    Columns("A:F").Select
    Selection.Sort Key1:=range("A1"), Order1:=xlAscending, Key2:=range _
        ("C1"), Order2:=xlAscending, Header:=xlGuess, OrderCustom:=1, _
        MatchCase:=False, Orientation:=xlTopToBottom
    range("A1").Select
End Sub

'
'get_con returns connection string from MSQuery - left from testing
'DSN=CLDSUTEST;UID=cldstestid;SRVR=dante;DB=CLDS_USER_DB;PWD=johnray1
'
Sub get_con()
  chan = Application.DDEInitiate("MSQuery", "Query1")
  connect = Application.DDERequest(chan, "ConnectionString")
  Sheets("Status").range("a1") = connect
  Application.DDETerminate chan
End Sub

'Sub trimcell() 'left from testing
'  tmp$ = Sheets("sheet1").Range("a1")
'  dashloc% = InStr(tmp$, "-")
'  tmp$ = Left$(tmp$, dashloc% - 1)
'  Sheets("sheet1").Range("a1") = tmp$
'End Sub

Sub close_conn()   ' used during testing to ensure all connections to Sybase are closed after failure
  For k% = 1 To 5
    sqlclose (k%)
  Next k%
End Sub

Sub back_to_data()   ' code for "Back to Query Sheet" button
  Sheets("Data").Select
  'ensure top row is visible
  Sheets("Data").range("a6").Select
  Sheets("Data").range("b1").Select
End Sub

Sub first50()    ' code for "First 50" button
  Sheets("First 50").Select
End Sub

Sub pending()    ' code for "Pending" button
  Sheets("Pending").Select
End Sub
```

```
Sub in_progress()    ' code for "In Progress" button
  Sheets("In Progress").Select
End Sub

Sub overview()    ' code for "Overview" button
  Sheets("Overview").Select
End Sub

Sub clear_sort()  ' clear Sorted Data sheet
  If Sheets("Data").range("l1") > 0 Then
    s_clearange$ = "a1:f" & Trim(Str$(Sheets("Data").range("l1")))
  Else
    s_clearange$ = "a1:f2500"
  End If
  Sheets("Sorted Data").range(s_clearange$).ClearContents
  Worksheets("Sorted Data").Calculate
  Sheets("Data").range("b1").Select
End Sub

Sub clear_data()  ' clear Data sheet
  If Sheets("Data").range("i1") > 0 Then
    clearange$ = "a6:j" & Trim(Str$(Sheets("Data").range("l1") + 5))
  Else
    clearange$ = "a6:j2500"
  End If
  Sheets("Data").range(clearange$).ClearContents
  Sheets("Data").range("i4:j4").ClearContents
  If Sheets("Sorted Data").range("a1") = "" Then
    Sheets("Data").range("l1").ClearContents
  End If
  Worksheets("Counts").range("b1:b5").ClearContents
  Worksheets("Counts").range("c1:g1").ClearContents
  Worksheets("Counts").range("a9:z10").ClearContents
  Worksheets("Data").Calculate
  Sheets("Data").range("b2").ClearContents
  Sheets("Data").range("a6").Select
  Sheets("Data").range("b1").Select
End Sub

'get_today_click macro
Sub get_today()  ' code for Today button
  Dim wksht As Object
  Set wksht = Worksheets("Data")
  curtime$ = Time$
  curdate$ = Date$
  my_hour = Val(Left$(curtime$, 2))
  my_minute = Val(Mid$(curtime$, 4, 2))
  my_month = Val(Left$(curdate$, 2))
  my_day = Val(Mid$(curdate$, 4, 2))
  my_year = Val(Right$(curdate$, 4)) - 1996
  wksht.DropDowns("start_month").ListIndex = my_month
  wksht.DropDowns("start_day").ListIndex = my_day
  wksht.DropDowns("start_year").ListIndex = my_year
  wksht.DropDowns("start_hour").ListIndex = 1
  wksht.DropDowns("start_minute").ListIndex = 1
  wksht.DropDowns("end_month").ListIndex = my_month
  wksht.DropDowns("end_day").ListIndex = my_day
  wksht.DropDowns("end_year").ListIndex = my_year
  wksht.DropDowns("end_hour").ListIndex = my_hour + 1
  If my_minute = 0 Then  'can't have listindex of 0
    my_minute = 1
  End If
```

```
  wksht.DropDowns("end_minute").ListIndex = my_minute
  test01  'call main module
End Sub

Sub reset_count()  ' code for Reset button
  Dim wksht As Object
  Set wksht = Worksheets("Data")
  curdate$ = Date$
  my_month = Val(Left$(curdate$, 2))
  my_year = Val(Right$(curdate$, 4)) - 1996
  curtime$ = Time$
  my_hour = Val(Left$(curtime$, 2))
  my_minute = Val(Mid$(curtime$, 4, 2))
  wksht.DropDowns("start_month").ListIndex = my_month
  wksht.DropDowns("start_day").ListIndex = 1
  wksht.DropDowns("start_year").ListIndex = my_year
  wksht.DropDowns("start_hour").ListIndex = 1
  wksht.DropDowns("start_minute").ListIndex = 1
  wksht.DropDowns("end_month").ListIndex = my_month
  wksht.DropDowns("end_day").ListIndex = 1
  wksht.DropDowns("end_year").ListIndex = my_year
  wksht.DropDowns("end_hour").ListIndex = 1
  wksht.DropDowns("end_minute").ListIndex = 1
  wksht.range("a6").Select
  wksht.range("b1").Select
End Sub

Sub dayofweek()  ' code from testing. for future enhancements
  Dim days$(7)
  days$(1) = "Sunday"
  days$(2) = "Monday"
  days$(3) = "Tueday"
  days$(4) = "Wednesday"
  days$(5) = "Thursday"
  days$(6) = "Friday"
  days$(7) = "Saturday"
  theday% = WeekDay(Date)
'  If theday% = 6 Then
'    MsgBox "TGIF"
'  Else
'    MsgBox "Today is " & days$(theday%)
'  End If
'get first event date/time
  etime$ = Sheets("Data").range("c6")
'check for empty cell before processing
If etime$ = "" Then
'handle error
End If
'find delimiter
  spaceloc% = InStr(etime$, " ")
' MsgBox "Event day is " & days$(WeekDay(Sheets("Data").Range("c6")))
 'MsgBox "Hour is " & Hour(Mid$(etime$, spaceloc% + 1)) & " TimeValue is " & TimeValue(Mid$(etime$, spaceloc% + 1))
'get second evnt date/time
 stime$ = Sheets("Data").range("c7")
'find delimiter
 spaceloc1% = InStr(stime$, " ")
'compute time between events - if events bridge days, see note below on parsing
 MsgBox "Minutes = " & Minute(TimeValue(Mid$(stime$, spaceloc% + 1)) - TimeValue(Mid$(etime$, spaceloc1% + 1))),
64, , "CLDS Metrics Report"
 End Sub

Sub showlabel()
```

```
  Sheets("Data").Labels("Label 41").Visible = True
End Sub

Sub showtime()
  Sheets("Data").range("c8") = Hour(Now) & ":" & Minute(Now)
End Sub
```