Automated SalesLogix Integrity Checks

Overview:
SalesLogix currently has 107 different integrity checks that can be run from the Administrator program, but SalesLogix failed to provide a way to schedule them so they needed to be run manually. The automated version of these integrity checks will allow for the tests to be run as a scheduled SQL job and notify support personnel only when there is a problem. The 'SLX_Missions_IntegrityChecks' job has been scheduled to run every weekday at 5:00am.

Important – This job is only intended to identify problems and notify support personnel. Any problems must be fixed by running the integrity checks from within SalesLogix manually so any changes to the database will also be sent to the remote users.

The job runs each test and inserts a record in the 'dbo.LCMS_IntegrityChecks' table if the error count is greater than 0 for any of the tests. The record contains the name of the test and the number of errors. After all tests have been performed, the job sends an email notification with a list of tests that failed to support personnel if any records have been inserted into the 'LCMS_IntegrityChecks' table.

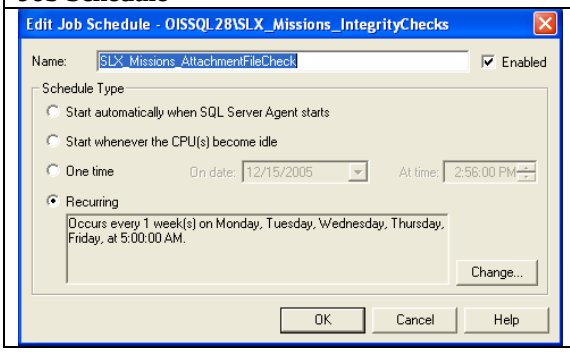
For all of the tests except 2, the SQL statement was able to be copied from SalesLogix and modified slightly, but the 'Attachment file validation' and the 'Library Docs file validation' tests require extra steps to determine if any files are missing. For these tests it is necessary to get a directory listing for the attachment and library files, then compare the entries in the 'Attachment' and 'LibraryDocs' tables to the list of files in the directories. Any file that is listed in the 'Attachment' or 'LibraryDocs' table is listed as an error if the file cannot also be found in the directory listing.

All logging information is sent to the \\oissql28\f$\IntegrityChecks|SLX_Missions\IntegrityCheck.log file.

How it Works:

1. Generate Attachment & Library File List – Creates text files (\\oissql28\f$\IntegrityChecks|SLX_Missions\Attachments.txt & \\oissql28\f$\IntegrityChecks|SLX_Missions\Library.txt) containing a bare list (/b) of all files, including subdirectories (/s) in the Documents and LibraryDocs directories.
2. SLX_Missions_AttachmentFileCheck – Transforms the data in the text file to a SQL table and parses out the Attachment file name that SalesLogix expects to see.
3. SLX_Missions_LibraryFileCheck – Transforms the data in the text file to a SQL table and parses out the Library Document file name that SalesLogix expects to see.
4. SLX Integrity Checks – Executes the stored procedure sp_LCMS_IntegrityChecks which clears the dbo.LCMS.IntegrityChecks table and inserts a record for each test that finds problems. The record contains the name of the test and the number of errors.
5. Send Email When Errors – Sends an email notification to support personnel if any records were inserted in the dbo.LCMS.IntegrityChecks table.

Job Schedule



Job Steps

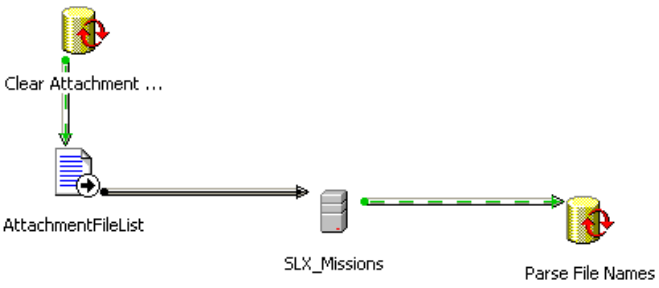
ID	Step Name	Type	On Success	On Failure
1	Generate Attachment & Library File List	Operating System Command	Goto next step	Quit with failure
2	SLX_Missions_AttachmentFileCheck	Operating System Command	Goto next step	Quit with failure
3	SLX_Missions_LibraryFileCheck	Operating System Command	Goto next step	Quit with failure
4	SLX Integrity Checks	Transact-SQL Script	Goto next step	Quit with failure
5	Send Email When Errors	Transact-SQL Script	Quit with success	Quit with failure

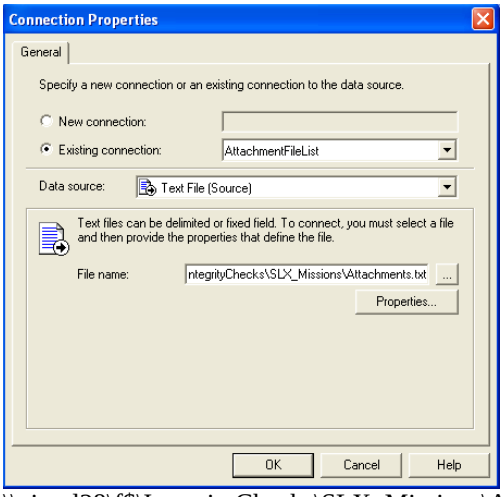
Step Name	Command
Generate Attachment & Library File List	f:\IntegrityChecks\SLX_Missions\ListFilesInDirs.bat
SLX_Missions_AttachmentFileCheck	DTSRun /S OISSQL28 /N SLX_Missions_AttachmentFileCheck /E
SLX_Missions_LibraryFileCheck	DTSRun /S OISSQL28 /N SLX_Missions_LibraryFileCheck /E
SLX Integrity Checks	exec sp_lcms_IntegrityChecks
Send Email When Errors	exec sp_LCMS_IntegrityCheck_SendEmail

ListFilesInDirs.bat (Batch file)

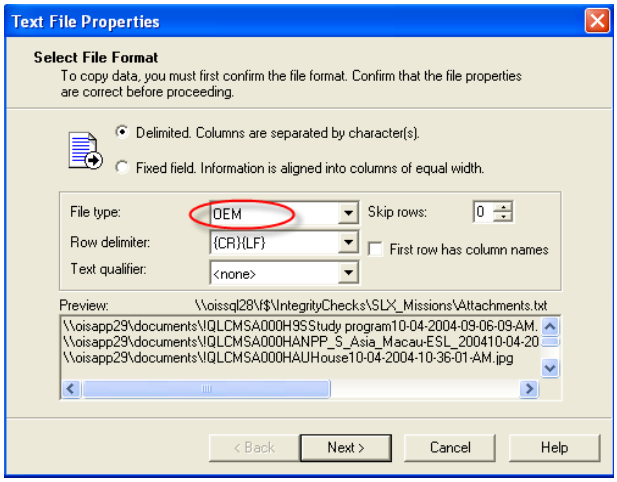
```
dir \\oisapp29\documents /b /s > "f:\IntegrityChecks\SLX_Missions\Attachments.txt"
dir \\oisapp29\library /b /s > "f:\IntegrityChecks\SLX_Missions\Library.txt"
```

SLX_Missions_AttachmentFileCheck (DTS package)

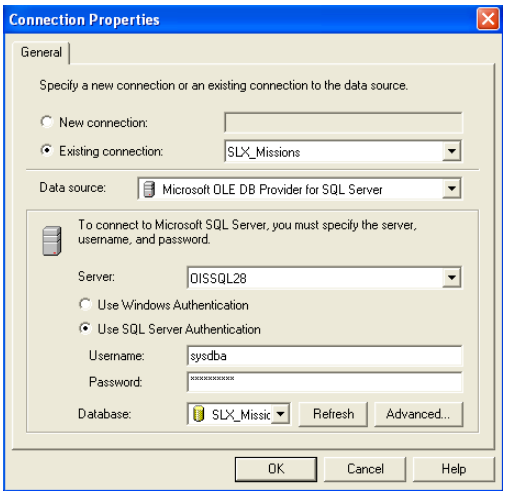


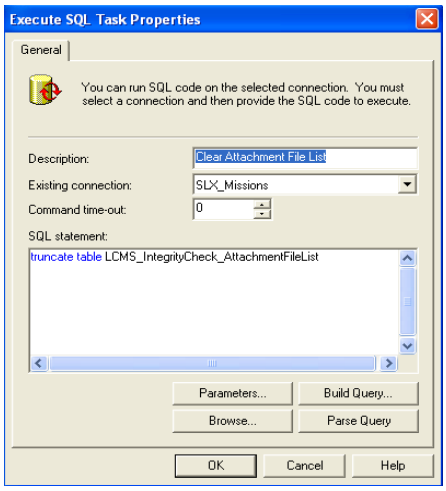


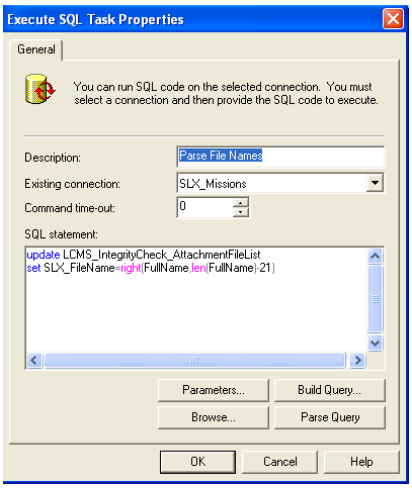
\\oissql28\f$\IntegrityChecks\SLX_Missions\Attachments.txt



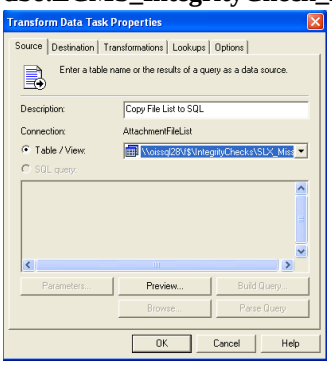
Be sure OEM is selected or documents with foreign characters (i.e. – Ĕ, ë, æ, ç, ï, ÿ, ç) in the name will be listed as missing when they aren't.

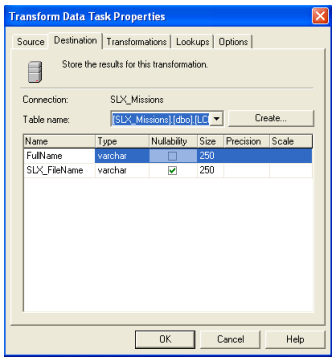


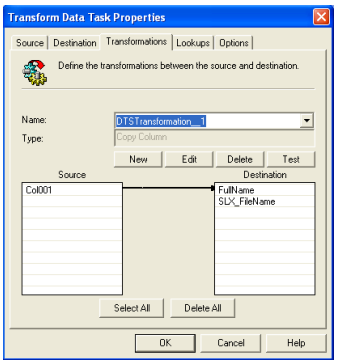


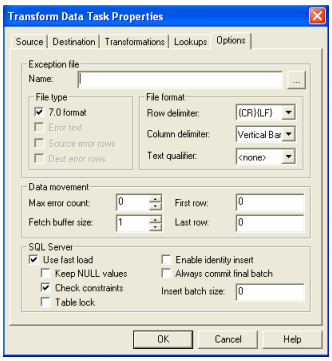


Data Transformation from [\\oissql28\f$\IntegrityChecks\SLX_Missions\Attachments.txt](\\oissql28\f$\IntegrityChecks\SLX_Missions\Attachments.txt) to dbo.LCMS_IntegrityCheck_AttachmentFileList









SLX_Missions_LibraryFileCheck (DTS package)

This DTS package is just like the SLX_Missions_AttachmentFileCheck DTS package (see above) except it uses the LibraryDocs table and the [\\oissql28\f$\IntegrityChecks\SLX_Missions\Library.txt](\\oissql28\f$\IntegrityChecks\SLX_Missions\Library.txt) file.

sp_LCMS_IntegrityChecks (Stored Procedure)

```
CREATE PROCEDURE [dbo].[sp_LCMS_IntegrityChecks] AS
-- SalesLogix Integrity Checker
-- Number of tests selected is 107
Declare @TestCount as int

-- >> Clear Results Table (LCMS_IntegrityChecks) <<
   truncate table LCMS_IntegrityChecks

-- >> Bad Account Mirrors (Uppercase) <<
   SELECT @TestCount=COUNT(ACCOUNTID) FROM SYSDBA.ACCOUNT
   WHERE ACCOUNT_UC <> UPPER(ACCOUNT) OR (ACCOUNT_UC IS NULL AND ACCOUNT IS NOT NULL)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Bad Account Mirrors (Uppercase) <<',@TestCount)

-- >> Missing Account Summary Records (Insert) <<
```

```
   SELECT @TestCount=COUNT(ACCOUNTID) FROM SYSDBA.ACCOUNT A1
   WHERE NOT EXISTS
         (SELECT ACCOUNTID FROM SYSDBA.ACCOUNTSUMMARY A2 WHERE A1.ACCOUNTID = A2.ACCOUNTID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Missing Account Summary Records (Insert) <<',@TestCount)

-- >> Account Summary with the wrong owner (Set to Account owner) <<
   SELECT @TestCount=COUNT(ACCOUNTID) FROM SYSDBA.ACCOUNTSUMMARY
   WHERE SECCODEID <>
         (SELECT SECCODEID FROM SYSDBA.ACCOUNT A2 WHERE SYSDBA.ACCOUNTSUMMARY.ACCOUNTID =
A2.ACCOUNTID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Account Summary with the wrong owner (Set to Account owner) <<',@TestCount)

-- >> Account Summary where Account has been deleted. (Delete) <<
   SELECT @TestCount=COUNT(ACCOUNTID) FROM SYSDBA.ACCOUNTSUMMARY
   WHERE NOT EXISTS
         (SELECT ACCOUNTID FROM SYSDBA.ACCOUNT A1
               WHERE SYSDBA.ACCOUNTSUMMARY.ACCOUNTID = A1.ACCOUNTID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Account Summary where Account has been deleted. (Delete) <<',@TestCount)

-- >> Accounts with missing Addresses (Created) <<
   SELECT @TestCount=COUNT(ACCOUNTID) FROM SYSDBA.ACCOUNT A1
   WHERE NOT EXISTS
         (SELECT ADDRESSID FROM SYSDBA.ADDRESS A2 WHERE A1.ADDRESSID = A2.ADDRESSID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Accounts with missing Addresses (Created) <<',@TestCount)

-- >> Accounts with missing Shipping Addresses (Set to Address) <<
   SELECT @TestCount=COUNT(ACCOUNTID) FROM SYSDBA.ACCOUNT
   WHERE NOT EXISTS
         (SELECT ADDRESSID FROM SYSDBA.ADDRESS A2
               WHERE SYSDBA.ACCOUNT.SHIPPINGID = A2.ADDRESSID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Accounts with missing Shipping Addresses (Set to Address) <<',@TestCount)

-- >> Accounts Not Owned by Anyone (Set to Everyone) <<
   SELECT @TestCount=COUNT(ACCOUNTID) FROM SYSDBA.ACCOUNT
   WHERE NOT EXISTS
         (SELECT SECCODEID FROM SYSDBA.SECCODE A2
               WHERE SYSDBA.ACCOUNT.SECCODEID=A2.SECCODEID)
         AND NOT EXISTS (SELECT USERID FROM SYSDBA.USERINFO A2
               WHERE SYSDBA.ACCOUNT.SECCODEID=A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Accounts Not Owned by Anyone (Set to Everyone) <<',@TestCount)
```

```
-- >> Accounts with bad Account Manager IDs (Clear) <<
   SELECT @TestCount=COUNT(ACCOUNTID) FROM SYSDBA.ACCOUNT
   WHERE (ACCOUNTMANAGERID IS NOT NULL) AND (ACCOUNTMANAGERID <> '')
         AND NOT EXISTS
         (SELECT USERID FROM SYSDBA.USERINFO A2
               WHERE SYSDBA.ACCOUNT.ACCOUNTMANAGERID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Accounts with bad Account Manager IDs (Clear) <<',@TestCount)

-- >> Activities with bad User IDs (Delete) <<
   SELECT @TestCount=COUNT(ACTIVITYID) FROM SYSDBA.ACTIVITY
   WHERE NOT EXISTS
         (SELECT USERID FROM SYSDBA.USERINFO A2
               WHERE SYSDBA.ACTIVITY.USERID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Activities with bad User IDs (Delete) <<',@TestCount)

-- >> Activities with bad Account IDs (Delete) <<
   SELECT @TestCount=COUNT(ACTIVITYID) FROM SYSDBA.ACTIVITY
   WHERE (ACCOUNTID IS NOT NULL) AND (ACCOUNTID <> '')
         AND NOT EXISTS
         (SELECT ACCOUNTID FROM SYSDBA.ACCOUNT A2
               WHERE SYSDBA.ACTIVITY.ACCOUNTID = A2.ACCOUNTID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Activities with bad Account IDs (Delete) <<',@TestCount)

-- >> Activities with bad Contact IDs (Delete) <<
   SELECT @TestCount=COUNT(ACTIVITYID) FROM SYSDBA.ACTIVITY
   WHERE (CONTACTID IS NOT NULL) AND (CONTACTID <> '')
         AND NOT EXISTS
         (SELECT CONTACTID FROM SYSDBA.CONTACT A2
               WHERE SYSDBA.ACTIVITY.CONTACTID = A2.CONTACTID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Activities with bad Contact IDs (Delete) <<',@TestCount)

-- >> Activities with bad Opportunity IDs (Delete) <<
   SELECT @TestCount=COUNT(ACTIVITYID) FROM SYSDBA.ACTIVITY
   WHERE (OPPORTUNITYID IS NOT NULL) AND (OPPORTUNITYID <> '')
         AND NOT EXISTS
         (SELECT OPPORTUNITYID FROM SYSDBA.OPPORTUNITY A2
               WHERE SYSDBA.ACTIVITY.OPPORTUNITYID = A2.OPPORTUNITYID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Activities with bad Opportunity IDs (Delete) <<',@TestCount)

-- >> User Activity Records with Bad User IDs (Delete) <<
   SELECT @TestCount=COUNT(ACTIVITYID) FROM SYSDBA.USER_ACTIVITY
   WHERE NOT EXISTS
```

```
        (SELECT USERID FROM SYSDBA.USERINFO A2
                WHERE SYSDBA.USER_ACTIVITY.USERID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> User Activity Records with Bad User IDs (Delete) <<',@TestCount)

-- >> User Activity Records with Bad Activity IDs (Delete) <<
   SELECT @TestCount=COUNT(ACTIVITYID) FROM SYSDBA.USER_ACTIVITY
   WHERE NOT EXISTS
        (SELECT ACTIVITYID FROM SYSDBA.ACTIVITY A2
                WHERE SYSDBA.USER_ACTIVITY.ACTIVITYID = A2.ACTIVITYID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> User Activity Records with Bad Activity IDs (Delete) <<',@TestCount)

-- >> Activities on No Calendar (Create) <<
   SELECT @TestCount=COUNT(ACTIVITYID) FROM SYSDBA.ACTIVITY A1
   WHERE NOT EXISTS
        (SELECT ACTIVITYID FROM SYSDBA.USER_ACTIVITY A2 WHERE A1.ACTIVITYID = A2.ACTIVITYID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> Activities on No Calendar (Create) <<',@TestCount)

-- >> User Notification Records with Bad From User IDs (Delete) <<
   SELECT @TestCount=COUNT(NOTIFYID) FROM SYSDBA.USERNOTIFICATION
   WHERE (FROMUSERID IS NOT NULL) AND (FROMUSERID <> '') AND NOT EXISTS
        (SELECT USERID FROM SYSDBA.USERINFO A2
                WHERE SYSDBA.USERNOTIFICATION.FROMUSERID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> User Notification Records with Bad From User IDs (Delete) <<',@TestCount)

-- >> User Notification Records with Bad To User IDs (Delete) <<
   SELECT @TestCount=COUNT(NOTIFYID) FROM SYSDBA.USERNOTIFICATION
   WHERE NOT EXISTS
        (SELECT USERID FROM SYSDBA.USERINFO A2
                WHERE SYSDBA.USERNOTIFICATION.TOUSERID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> User Notification Records with Bad To User IDs (Delete) <<',@TestCount)

-- >> Bad Contact LastName Mirrors (Uppercase) <<
   SELECT @TestCount=COUNT(CONTACTID) FROM SYSDBA.CONTACT
   WHERE LASTNAME_UC <> UPPER(LASTNAME) OR (LASTNAME_UC IS NULL AND LASTNAME IS NOT NULL)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> Bad Contact LastName Mirrors (Uppercase) <<',@TestCount)

-- >> Contacts with no Account (Delete) <<
   SELECT @TestCount=COUNT(CONTACTID) FROM SYSDBA.CONTACT
   WHERE NOT EXISTS
        (SELECT ACCOUNTID FROM SYSDBA.ACCOUNT A2
```

```
                WHERE SYSDBA.CONTACT.ACCOUNTID = A2.ACCOUNTID)

  IF @TestCount<>0
  Insert into LCMS_IntegrityChecks (TestName,TestCount)
      Values('>> Contacts with no Account (Delete) <<',@TestCount)

-- >> Contacts with missing Addresses (Created) <<
  SELECT @TestCount=COUNT(CONTACTID) FROM SYSDBA.CONTACT A1
  WHERE NOT EXISTS (SELECT ADDRESSID FROM SYSDBA.ADDRESS A2 WHERE A1.ADDRESSID = A2.ADDRESSID)

  IF @TestCount<>0
  Insert into LCMS_IntegrityChecks (TestName,TestCount)
      Values('>> Contacts with missing Addresses (Created) <<',@TestCount)

-- >> Contacts with missing Shipping Addresses (Set to Address) <<
  SELECT @TestCount=COUNT(CONTACTID) FROM SYSDBA.CONTACT
  WHERE NOT EXISTS
      (SELECT ADDRESSID FROM SYSDBA.ADDRESS A2
          WHERE SYSDBA.CONTACT.SHIPPINGID = A2.ADDRESSID)

  IF @TestCount<>0
  Insert into LCMS_IntegrityChecks (TestName,TestCount)
      Values('>> Contacts with missing Shipping Addresses (Set to Address) <<',@TestCount)

-- >> Contacts with the wrong owner (Set to Account owner) <<
  SELECT @TestCount=COUNT(CONTACTID) FROM SYSDBA.CONTACT
  WHERE SECCODEID <>
      (SELECT SECCODEID FROM SYSDBA.ACCOUNT A2
          WHERE SYSDBA.CONTACT.ACCOUNTID = A2.ACCOUNTID)

  IF @TestCount<>0
  Insert into LCMS_IntegrityChecks (TestName,TestCount)
      Values('>> Contacts with the wrong owner (Set to Account owner) <<',@TestCount)

-- >> Contacts with bad Account Manager IDs (Clear) <<
  SELECT @TestCount=COUNT(CONTACTID) FROM SYSDBA.CONTACT
  WHERE (ACCOUNTMANAGERID IS NOT NULL) AND (ACCOUNTMANAGERID <> '') AND NOT EXISTS
      (SELECT USERID FROM SYSDBA.USERINFO A2
          WHERE SYSDBA.CONTACT.ACCOUNTMANAGERID = A2.USERID)

  IF @TestCount<>0
  Insert into LCMS_IntegrityChecks (TestName,TestCount)
      Values('>> Contacts with bad Account Manager IDs (Clear) <<',@TestCount)

-- >> Contact_LeadSource: Unattached to Contacts (Delete) <<
  SELECT @TestCount=COUNT(CONTACTID) FROM SYSDBA.CONTACT_LEADSOURCE
  WHERE NOT EXISTS
      (SELECT CONTACTID FROM SYSDBA.CONTACT A2
          WHERE SYSDBA.CONTACT_LEADSOURCE.CONTACTID = A2.CONTACTID)

  IF @TestCount<>0
  Insert into LCMS_IntegrityChecks (TestName,TestCount)
      Values('>> Contact_LeadSource: Unattached to Contacts (Delete) <<',@TestCount)

-- >> Contact_LeadSource: Unattached to Lead Sources (Delete) <<
  SELECT @TestCount=COUNT(CONTACTID) FROM SYSDBA.CONTACT_LEADSOURCE
  WHERE NOT EXISTS
      (SELECT LEADSOURCEID FROM SYSDBA.LEADSOURCE A2
```

```
                WHERE SYSDBA.CONTACT_LEADSOURCE.LEADSOURCEID = A2.LEADSOURCEID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Contact_LeadSource: Unattached to Lead Sources (Delete) <<',@TestCount)

-- >> Contact link to Account is broken. (Fix) <<
   SELECT @TestCount=COUNT(CONTACTID) FROM SYSDBA.CONTACT
   WHERE (ACCOUNT <>
         (SELECT ACCOUNT FROM SYSDBA.ACCOUNT A2
                WHERE SYSDBA.CONTACT.ACCOUNTID = A2.ACCOUNTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Contact link to Account is broken. (Fix) <<',@TestCount)

-- >> 04-01: Contracts with invalid Account ID (Delete Contract) <<
   SELECT @TestCount=COUNT(CONTRACTID) FROM SYSDBA.CONTRACT
   WHERE (ACCOUNTID IS NULL) OR (NOT EXISTS
         (SELECT B.ACCOUNTID FROM SYSDBA.ACCOUNT B
                WHERE B.ACCOUNTID = SYSDBA.CONTRACT.ACCOUNTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 04-01: Contracts with invalid Account ID (Delete Contract) <<',@TestCount)

-- >> 04-02: Contract Incidents with invalid Contract ID (Delete ContractIncident) <<
   SELECT @TestCount=COUNT(CONTRACTID) FROM SYSDBA.CONTRACTINCIDENT
   WHERE (CONTRACTID IS NULL) OR (NOT EXISTS
         (SELECT B.CONTRACTID FROM SYSDBA.CONTRACT B
                WHERE B.CONTRACTID = SYSDBA.CONTRACTINCIDENT.CONTRACTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 04-02: Contract Incidents with invalid Contract ID (Delete ContractIncident) <<',@TestCount)

-- >> 04-03: Contract Items with invalid Contract ID (Delete ContractItem) <<
   SELECT @TestCount=COUNT(CONTRACTID) FROM SYSDBA.CONTRACTITEM
   WHERE (CONTRACTID IS NULL) OR (NOT EXISTS
         (SELECT B.CONTRACTID FROM SYSDBA.CONTRACT B
                WHERE B.CONTRACTID = SYSDBA.CONTRACTITEM.CONTRACTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 04-03: Contract Items with invalid Contract ID (Delete ContractItem) <<',@TestCount)

-- >> 05-01: Defect Ticket cross references with invalid Ticket ID or Defect ID (Delete DefectTicket) <<
   SELECT @TestCount=COUNT(DEFECTTICKETID) FROM SYSDBA.DEFECTTICKET
   WHERE (NOT EXISTS (SELECT B.TICKETID FROM SYSDBA.TICKET B
                WHERE B.TICKETID = SYSDBA.DEFECTTICKET.TICKETID))
   OR (NOT EXISTS (SELECT C.DEFECTID FROM SYSDBA.DEFECT C
                WHERE C.DEFECTID = SYSDBA.DEFECTTICKET.DEFECTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 05-01: Defect Ticket cross references with invalid Ticket ID or Defect ID (Delete DefectTicket) <<',@TestCount)

-- >> 05-02: Defects with invalid RecordedBy ID (Clear RecordedByID) <<
```

```
   SELECT @TestCount=COUNT(DEFECTID) FROM SYSDBA.DEFECT
   WHERE (RECORDEDBYID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.SECCODEID FROM SYSDBA.SECCODE B
               WHERE B.SECCODEID = SYSDBA.DEFECT.RECORDEDBYID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 05-02: Defects with invalid RecordedBy ID (Clear RecordedByID) <<',@TestCount)

-- >> 05-03: Defects with invalid ClosedBy ID (Clear ClosedByID) <<
   SELECT @TestCount=COUNT(DEFECTID) FROM SYSDBA.DEFECT
   WHERE (CLOSEDBYID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.SECCODEID FROM SYSDBA.SECCODE B
               WHERE B.SECCODEID = SYSDBA.DEFECT.CLOSEDBYID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 05-03: Defects with invalid ClosedBy ID (Clear ClosedByID) <<',@TestCount)

-- >> 05-04: Defects with invalid owner (Set to Everyone) <<
   SELECT @TestCount=COUNT(DEFECTID) FROM SYSDBA.DEFECT
   WHERE (SECCODEID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.SECCODEID FROM SYSDBA.SECCODE B
               WHERE B.SECCODEID = SYSDBA.DEFECT.SECCODEID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 05-04: Defects with invalid owner (Set to Everyone) <<',@TestCount)

-- >> 05-05: Defect Histories with invalid Defect ID (Delete DefectHistory) <<
   SELECT @TestCount=COUNT(DEFECTHISID) FROM SYSDBA.DEFECTHISTORY
   WHERE (DEFECTID IS NULL) OR (NOT EXISTS
         (SELECT B.DEFECTID FROM SYSDBA.DEFECT B WHERE B.DEFECTID =
               SYSDBA.DEFECTHISTORY.DEFECTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 05-05: Defect Histories with invalid Defect ID (Delete DefectHistory) <<',@TestCount)

-- >> 05-06: Defect Problems with invalid Defect ID (Delete DefectProblem) <<
   SELECT @TestCount=COUNT(DEFECTID) FROM SYSDBA.DEFECTPROBLEM
   WHERE (DEFECTID IS NULL) OR (NOT EXISTS
         (SELECT B.DEFECTID FROM SYSDBA.DEFECT B
               WHERE B.DEFECTID = SYSDBA.DEFECTPROBLEM.DEFECTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 05-06: Defect Problems with invalid Defect ID (Delete DefectProblem) <<',@TestCount)

-- >> 05-07: Defect Solutions with invalid Defect ID (Delete DefectSolution) <<
   SELECT @TestCount=COUNT(DEFECTID) FROM SYSDBA.DEFECTSOLUTION
   WHERE (DEFECTID IS NULL) OR (NOT EXISTS
         (SELECT B.DEFECTID FROM SYSDBA.DEFECT B
               WHERE B.DEFECTID = SYSDBA.DEFECTSOLUTION.DEFECTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 05-07: Defect Solutions with invalid Defect ID (Delete DefectSolution) <<',@TestCount)
```

```
-- >> Histories with bad User IDs (Clear) <<
   SELECT @TestCount=COUNT(HISTORYID) FROM SYSDBA.HISTORY
   WHERE (USERID IS NOT NULL) AND (USERID <> '') AND NOT EXISTS
         (SELECT USERID FROM SYSDBA.USERINFO A2 WHERE SYSDBA.HISTORY.USERID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Histories with bad User IDs (Clear) <<',@TestCount)

-- >> Histories with bad Account IDs (Clear) <<
   SELECT @TestCount=COUNT(HISTORYID) FROM SYSDBA.HISTORY
   WHERE (ACCOUNTID IS NOT NULL) AND (ACCOUNTID <> '') AND NOT EXISTS
         (SELECT ACCOUNTID FROM SYSDBA.ACCOUNT A2
               WHERE SYSDBA.HISTORY.ACCOUNTID = A2.ACCOUNTID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Histories with bad Account IDs (Clear) <<',@TestCount)

-- >> Histories with bad Contact IDs (Clear) <<
   SELECT @TestCount=COUNT(HISTORYID) FROM SYSDBA.HISTORY
   WHERE (CONTACTID IS NOT NULL) AND (CONTACTID <> '') AND NOT EXISTS
         (SELECT CONTACTID FROM SYSDBA.CONTACT A2
               WHERE SYSDBA.HISTORY.CONTACTID = A2.CONTACTID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Histories with bad Contact IDs (Clear) <<',@TestCount)

-- >> Histories with bad Opportunity IDs (Clear) <<
   SELECT @TestCount=COUNT(HISTORYID) FROM SYSDBA.HISTORY
   WHERE (OPPORTUNITYID IS NOT NULL) AND (OPPORTUNITYID <> '') AND NOT EXISTS
         (SELECT OPPORTUNITYID FROM SYSDBA.OPPORTUNITY A2
               WHERE SYSDBA.HISTORY.OPPORTUNITYID = A2.OPPORTUNITYID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Histories with bad Opportunity IDs (Clear) <<',@TestCount)

-- >> Opportunities with no Account (Delete) <<
   SELECT @TestCount=COUNT(OPPORTUNITYID) FROM SYSDBA.OPPORTUNITY
   WHERE NOT EXISTS (SELECT ACCOUNTID FROM SYSDBA.ACCOUNT A2
               WHERE SYSDBA.OPPORTUNITY.ACCOUNTID = A2.ACCOUNTID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Opportunities with no Account (Delete) <<',@TestCount)

-- >> Opportunities with the wrong owner (Set to Account owner) <<
   SELECT @TestCount=COUNT(OPPORTUNITYID) FROM SYSDBA.OPPORTUNITY
   WHERE SECCODEID <> (SELECT SECCODEID FROM SYSDBA.ACCOUNT A2
               WHERE SYSDBA.OPPORTUNITY.ACCOUNTID = A2.ACCOUNTID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Opportunities with the wrong owner (Set to Account owner) <<',@TestCount)
```

Automated SalesLogix Integrity Checks

```
-- >> Opportunities with bad Account Manager IDs (Clear) <<
   SELECT @TestCount=COUNT(OPPORTUNITYID) FROM SYSDBA.OPPORTUNITY
   WHERE (ACCOUNTMANAGERID IS NOT NULL) AND (ACCOUNTMANAGERID <> '') AND NOT EXISTS
         (SELECT USERID FROM SYSDBA.USERINFO A2
               WHERE SYSDBA.OPPORTUNITY.ACCOUNTMANAGERID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Opportunities with bad Account Manager IDs (Clear) <<',@TestCount)

-- >> Opportunities with bad LeadSources (Clear) <<
   SELECT @TestCount=COUNT(OPPORTUNITYID) FROM SYSDBA.OPPORTUNITY
   WHERE (LEADSOURCEID IS NOT NULL) AND (LEADSOURCEID <> '') AND NOT EXISTS
         (SELECT LEADSOURCEID FROM SYSDBA.LEADSOURCE A2
               WHERE SYSDBA.OPPORTUNITY.LEADSOURCEID = A2.LEADSOURCEID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Opportunities with bad LeadSources (Clear) <<',@TestCount)

-- >> Opportuntity_Contact: Unattached to Opportunities (Delete) <<
   SELECT @TestCount=COUNT(OPPCONTACTID) FROM SYSDBA.OPPORTUNITY_CONTACT
   WHERE NOT EXISTS
         (SELECT OPPORTUNITYID FROM SYSDBA.OPPORTUNITY A2
               WHERE SYSDBA.OPPORTUNITY_CONTACT.OPPORTUNITYID = A2.OPPORTUNITYID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Opportuntity_Contact: Unattached to Opportunities (Delete) <<',@TestCount)

-- >> Opportuntity_Contact: Unattached to Contacts (Delete) <<
   SELECT @TestCount=COUNT(OPPCONTACTID) FROM SYSDBA.OPPORTUNITY_CONTACT
   WHERE NOT EXISTS
         (SELECT CONTACTID FROM SYSDBA.CONTACT A2
               WHERE SYSDBA.OPPORTUNITY_CONTACT.CONTACTID = A2.CONTACTID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Opportuntity_Contact: Unattached to Contacts (Delete) <<',@TestCount)

-- >> Opportuntity_Competitor: Unattached to Opportunities (Delete) <<
   SELECT @TestCount=COUNT(OPPORTUNITYID) FROM SYSDBA.OPPORTUNITY_COMPETITOR
   WHERE NOT EXISTS
         (SELECT OPPORTUNITYID FROM SYSDBA.OPPORTUNITY A2
               WHERE SYSDBA.OPPORTUNITY_COMPETITOR.OPPORTUNITYID = A2.OPPORTUNITYID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Opportuntity_Competitor: Unattached to Opportunities (Delete) <<',@TestCount)

-- >> Opportuntity_Competitor: Unattached to Competitors (Delete) <<
   SELECT @TestCount=COUNT(OPPORTUNITYID) FROM SYSDBA.OPPORTUNITY_COMPETITOR
   WHERE NOT EXISTS
         (SELECT COMPETITORID FROM SYSDBA.COMPETITOR A2
               WHERE SYSDBA.OPPORTUNITY_COMPETITOR.COMPETITORID = A2.COMPETITORID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
```

```
        Values('>> Opportuntity_Competitor: Unattached to Competitors (Delete) <<',@TestCount)

-- >> Opportuntity_Product: Unattached to Opportunities (Delete) <<
   SELECT @TestCount=COUNT(OPPPRODUCTID) FROM SYSDBA.OPPORTUNITY_PRODUCT
   WHERE NOT EXISTS
        (SELECT OPPORTUNITYID FROM SYSDBA.OPPORTUNITY A2
                WHERE SYSDBA.OPPORTUNITY_PRODUCT.OPPORTUNITYID = A2.OPPORTUNITYID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> Opportuntity_Product: Unattached to Opportunities (Delete) <<',@TestCount)

-- >> Opportuntity_Product: Unattached to Products (Delete) <<
   SELECT @TestCount=COUNT(OPPPRODUCTID) FROM SYSDBA.OPPORTUNITY_PRODUCT
   WHERE NOT EXISTS
        (SELECT PRODUCTID FROM SYSDBA.PRODUCT A2
                WHERE SYSDBA.OPPORTUNITY_PRODUCT.PRODUCTID = A2.PRODUCTID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> Opportuntity_Product: Unattached to Products (Delete) <<',@TestCount)

-- >> Library Docs with bad Dir IDs (Delete) <<
   SELECT @TestCount=COUNT(FILEID) FROM SYSDBA.LIBRARYDOCS
   WHERE NOT EXISTS (SELECT DIRID FROM SYSDBA.LIBRARYDIRS A2
                WHERE SYSDBA.LIBRARYDOCS.DIRID = A2.DIRID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> Library Docs with bad Dir IDs (Delete) <<',@TestCount)


-- >> Library Docs file validation. (Delete) <<
   select @TestCount=count([FileName])
   from sysdba.LibraryDocs
   where [FileName] not in
   (Select SLX_FileName from dbo.lcms_IntegrityCheck_LibraryFileList)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> Library Docs file validation. (Delete) <<',@TestCount)

-- >> Attachment file validation. (Delete) <<
   --SELECT ATTACHID, DESCRIPTION, FILENAME FROM SYSDBA.ATTACHMENT
   select @TestCount=count([FileName])
   from sysdba.attachment
   where [FileName] not in
   (Select SLX_FileName from dbo.lcms_IntegrityCheck_AttachmentFileList)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> Attachment file validation. (Delete) <<',@TestCount)


-- >> Joins are invalid. (Delete) <<
   SELECT @TestCount=COUNT(JOINID) FROM SYSDBA.JOINDATA
   WHERE (FROMTABLE <> '*') AND(NOT (UPPER(FROMTABLE) IN
        (SELECT UPPER(TABLENAME) FROM SYSDBA.RESYNCTABLEDEFS)))
```

```
        OR (FROMTABLE <> '*') AND (NOT (UPPER(TOTABLE) IN
            (SELECT UPPER(TABLENAME) FROM SYSDBA.RESYNCTABLEDEFS)))

  IF @TestCount<>0
  Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> Joins are invalid. (Delete) <<',@TestCount)

-- >> 09-01: Products with invalid Vendor ID (Clear the VendorID) <<
  SELECT @TestCount=COUNT(PRODUCTID) FROM SYSDBA.PRODUCT
  WHERE (VENDOR IS NOT NULL) AND (NOT EXISTS
        (SELECT B.ACCOUNTID FROM SYSDBA.ACCOUNT B
            WHERE B.ACCOUNTID = SYSDBA.PRODUCT.VENDOR))

  IF @TestCount<>0
  Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> 09-01: Products with invalid Vendor ID (Clear the VendorID) <<',@TestCount)

-- >> 09-02: Account Products with invalid Account ID (Delete the AccountProduct) <<
  SELECT @TestCount=COUNT(ACCOUNTPRODUCTID) FROM SYSDBA.ACCOUNTPRODUCT
  WHERE (ACCOUNTID IS NULL) OR (NOT EXISTS
        (SELECT B.ACCOUNTID FROM SYSDBA.ACCOUNT B
            WHERE B.ACCOUNTID = SYSDBA.ACCOUNTPRODUCT.ACCOUNTID))

  IF @TestCount<>0
  Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('09-02: Account Products with invalid Account ID (Delete the AccountProduct) <<',@TestCount)

-- >> 09-03: Account Products with invalid Product ID (Delete the AccountProduct) <<
  SELECT @TestCount=COUNT(ACCOUNTPRODUCTID) FROM SYSDBA.ACCOUNTPRODUCT
  WHERE (PRODUCTID IS NULL) OR (NOT EXISTS
        (SELECT B.PRODUCTID FROM SYSDBA.PRODUCT B
            WHERE B.PRODUCTID = SYSDBA.ACCOUNTPRODUCT.PRODUCTID))

  IF @TestCount<>0
  Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> 09-03: Account Products with invalid Product ID (Delete the AccountProduct) <<',@TestCount)

-- >> 09-04: Account Products with invalid Account Product Parent ID (Clear the AccountProductParentID <<
  SELECT @TestCount=COUNT(ACCOUNTPRODUCTID) FROM SYSDBA.ACCOUNTPRODUCT
  WHERE (ACCOUNTPRODUCTPARENTID IS NOT NULL) AND (NOT EXISTS
        (SELECT B.ACCOUNTPRODUCTID FROM SYSDBA.ACCOUNTPRODUCT B
            WHERE B.ACCOUNTPRODUCTID = SYSDBA.ACCOUNTPRODUCT.ACCOUNTPRODUCTPARENTID))

  IF @TestCount<>0
  Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> 09-04: Account Products with invalid Account Product Parent ID (Clear the AccountProductParentID
<<',@TestCount)

-- >> 09-05: Defect Products with invalid Product ID (Delete the DefectProduct) <<
  SELECT @TestCount=COUNT(DEFECTID) FROM SYSDBA.DEFECTPRODUCT
  WHERE (PRODUCTID IS NULL) OR (NOT EXISTS
        (SELECT B.PRODUCTID FROM SYSDBA.PRODUCT B
            WHERE B.PRODUCTID = SYSDBA.DEFECTPRODUCT.PRODUCTID))

  IF @TestCount<>0
  Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> 09-05: Defect Products with invalid Product ID (Delete the DefectProduct) <<',@TestCount)
```

Automated SalesLogix Integrity Checks

```
-- >> 09-06: Defect Products with invalid Defect ID (Delete the DefectProduct) <<
   SELECT @TestCount=COUNT(PRODUCTID) FROM SYSDBA.DEFECTPRODUCT
   WHERE (DEFECTID IS NULL) OR (NOT EXISTS
         (SELECT B.DEFECTID FROM SYSDBA.DEFECT B
               WHERE B.DEFECTID = SYSDBA.DEFECTPRODUCT.DEFECTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 09-06: Defect Products with invalid Defect ID (Delete the DefectProduct) <<',@TestCount)

-- >> 09-07: Ticket Account Products with invalid Ticket ID (Delete the TicketAccountProduct) <<
   SELECT @TestCount=COUNT(TICKETACCOUNTPRODUCTID) FROM SYSDBA.TICKETACCOUNTPRODUCT
   WHERE (TICKETID IS NULL) OR (NOT EXISTS
         (SELECT B.TICKETID FROM SYSDBA.TICKET B
               WHERE B.TICKETID = SYSDBA.TICKETACCOUNTPRODUCT.TICKETID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 09-07: Ticket Account Products with invalid Ticket ID (Delete the TicketAccountProduct) <<',@TestCount)

-- >> 09-08: Ticket Account Products with invalid AccountProduct ID (Delete the TicketAccountProduct) <<
   SELECT @TestCount=COUNT(TICKETACCOUNTPRODUCTID) FROM SYSDBA.TICKETACCOUNTPRODUCT
   WHERE (ACCOUNTPRODUCTID IS NULL) OR (NOT EXISTS
         (SELECT B.ACCOUNTPRODUCTID FROM SYSDBA.ACCOUNTPRODUCT B
               WHERE B.ACCOUNTPRODUCTID = SYSDBA.TICKETACCOUNTPRODUCT.ACCOUNTPRODUCTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 09-08: Ticket Account Products with invalid AccountProduct ID (Delete the TicketAccountProduct)
<<',@TestCount)

-- >> 10-02: RMAs with invalid Account ID and invalid Ticket Account ID (Delete RMA) <<
   SELECT @TestCount=COUNT(RMAID) FROM SYSDBA.RMA
   WHERE ((ACCOUNTID IS NULL) OR (NOT EXISTS
         (SELECT B.ACCOUNTID FROM SYSDBA.ACCOUNT B WHERE B.ACCOUNTID = SYSDBA.RMA.ACCOUNTID)))
         AND ((TICKETID IS NULL) OR (NOT EXISTS
               (SELECT C.TICKETID FROM SYSDBA.TICKET C
                     WHERE C.TICKETID = SYSDBA.RMA.TICKETID)))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 10-02: RMAs with invalid Account ID and invalid Ticket Account ID (Delete RMA) <<',@TestCount)

-- >> 10-01: RMAs with invalid Account ID (Fix Account with Ticket Account ID) <<
   SELECT @TestCount=COUNT(RMAID) FROM SYSDBA.RMA
   WHERE (ACCOUNTID IS NULL) OR (NOT EXISTS
         (SELECT B.ACCOUNTID FROM SYSDBA.ACCOUNT B WHERE B.ACCOUNTID = SYSDBA.RMA.ACCOUNTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 10-01: RMAs with invalid Account ID (Fix Account with Ticket Account ID) <<',@TestCount)

-- >> 10-03: RMAs with invalid SubmittedBy ID (Clear SubmittedByID) <<
   SELECT @TestCount=COUNT(RMAID) FROM SYSDBA.RMA
   WHERE (SUBMITTEDBY IS NOT NULL) AND (NOT EXISTS
         (SELECT B.SECCODEID FROM SYSDBA.SECCODE B WHERE B.SECCODEID = SYSDBA.RMA.SUBMITTEDBY))

   IF @TestCount<>0
```

```
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 10-03: RMAs with invalid SubmittedBy ID (Clear SubmittedByID) <<',@TestCount)

-- >> 10-04: RMAs with invalid Ticket ID (Clear TicketID) <<
   SELECT @TestCount=COUNT(RMAID) FROM SYSDBA.RMA
   WHERE (TICKETID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.TICKETID FROM SYSDBA.TICKET B WHERE B.TICKETID = SYSDBA.RMA.TICKETID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 10-04: RMAs with invalid Ticket ID (Clear TicketID) <<',@TestCount)

-- >> 10-05: RMAs with invalid ShipToAddress ID (Clear ShipToAddressID) <<
   SELECT @TestCount=COUNT(RMAID) FROM SYSDBA.RMA
   WHERE (SHIPTOADDRESSID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.ADDRESSID FROM SYSDBA.ADDRESS B
               WHERE B.ADDRESSID = SYSDBA.RMA.SHIPTOADDRESSID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 10-05: RMAs with invalid ShipToAddress ID (Clear ShipToAddressID) <<',@TestCount)

-- >> 10-06: RMAs with invalid ReturnByAddress ID (Clear ReturnByAddressID) <<
   SELECT @TestCount=COUNT(RMAID) FROM SYSDBA.RMA
   WHERE (RETURNBYADDRESSID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.ADDRESSID FROM SYSDBA.ADDRESS B
               WHERE B.ADDRESSID = SYSDBA.RMA.RETURNBYADDRESSID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 10-06: RMAs with invalid ReturnByAddress ID (Clear ReturnByAddressID) <<',@TestCount)

-- >> 10-07: RMAs with invalid ShipToContact ID (Clear ShipToContactID) <<
   SELECT @TestCount=COUNT(RMAID) FROM SYSDBA.RMA
   WHERE (SHIPTOCONTACTID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.CONTACTID FROM SYSDBA.CONTACT B
               WHERE B.CONTACTID = SYSDBA.RMA.SHIPTOCONTACTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 10-07: RMAs with invalid ShipToContact ID (Clear ShipToContactID) <<',@TestCount)

-- >> 10-08: RMAs with invalid ReturnedByContact ID (Clear ReturnedByContactID) <<
   SELECT @TestCount=COUNT(RMAID) FROM SYSDBA.RMA
   WHERE (RETURNEDBYCONTACTID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.CONTACTID FROM SYSDBA.CONTACT B
               WHERE B.CONTACTID = SYSDBA.RMA.RETURNEDBYCONTACTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 10-08: RMAs with invalid ReturnedByContact ID (Clear ReturnedByContactID) <<',@TestCount)

-- >> 10-09: RMA Received Products with invalid RMA ID (Delete RMAReceivedProduct) <<
   SELECT @TestCount=COUNT(RMAID) FROM SYSDBA.RMARECEIVEDPRODUCT
   WHERE (RMAID IS NULL) OR (NOT EXISTS
         (SELECT B.RMAID FROM SYSDBA.RMA B WHERE B.RMAID = SYSDBA.RMARECEIVEDPRODUCT.RMAID))

   IF @TestCount<>0
```

```
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 10-09: RMA Received Products with invalid RMA ID (Delete RMAReceivedProduct) <<',@TestCount)

-- >> 10-10: RMA Shipped Products with invalid RMA ID (Delete RMAShippedProduct) <<
   SELECT @TestCount=COUNT(RMAID) FROM SYSDBA.RMASHIPPEDPRODUCT
   WHERE (RMAID IS NULL) OR (NOT EXISTS
         (SELECT B.RMAID FROM SYSDBA.RMA B WHERE B.RMAID = SYSDBA.RMASHIPPEDPRODUCT.RMAID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 10-10: RMA Shipped Products with invalid RMA ID (Delete RMAShippedProduct) <<',@TestCount)

-- >> Sec Rights Records with invalid Sec Codes (Delete) <<
   SELECT @TestCount=COUNT(SECRIGHTSID) FROM SYSDBA.SECRIGHTS
   WHERE NOT EXISTS (SELECT SECCODEID FROM SYSDBA.SECCODE A2
         WHERE SYSDBA.SECRIGHTS.SECCODEID = A2.SECCODEID)
   AND NOT EXISTS (SELECT USERID FROM SYSDBA.USERINFO A2
         WHERE SYSDBA.SECRIGHTS.SECCODEID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Sec Rights Records with invalid Sec Codes (Delete) <<',@TestCount)

-- >> Sec Rights Records with invalid User IDs (Delete) <<
   SELECT @TestCount=COUNT(SECRIGHTSID) FROM SYSDBA.SECRIGHTS
   WHERE NOT EXISTS (SELECT USERID FROM SYSDBA.USERINFO A2
         WHERE SYSDBA.SECRIGHTS.ACCESSID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Sec Rights Records with invalid User IDs (Delete) <<',@TestCount)

-- >> Sec Rights Records with invalid Profile IDs (Delete) <<
   SELECT @TestCount=COUNT(SECRIGHTSID) FROM SYSDBA.SECRIGHTS
   WHERE NOT EXISTS
         (SELECT PROFILEID FROM SYSDBA.SECPROFILE A2
               WHERE SYSDBA.SECRIGHTS.PROFILEID = A2.PROFILEID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Sec Rights Records with invalid Profile IDs (Delete) <<',@TestCount)

-- >> Unused Profile Records (Delete) <<
   SELECT @TestCount=COUNT(PROFILEID) FROM SYSDBA.SECPROFILE
   WHERE NOT EXISTS
         (SELECT PROFILEID FROM SYSDBA.SECRIGHTS A2
               WHERE SYSDBA.SECPROFILE.PROFILEID = A2.PROFILEID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> Unused Profile Records (Delete) <<',@TestCount)

-- >> 12-01: Tickets with no Account ID (Fix Account with Contact Account ID) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE ((ACCOUNTID IS NULL) AND (CONTACTID IS NOT NULL))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
```

```
        Values('>> 12-01: Tickets with no Account ID (Fix Account with Contact Account ID) <<',@TestCount)

-- >> 12-02: Tickets with a different Account and Contact Account ID (Fix Account with Contact Account ID) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE ((ACCOUNTID IS NOT NULL) AND (CONTACTID IS NOT NULL)) AND (ACCOUNTID <>
        (SELECT B.ACCOUNTID FROM SYSDBA.CONTACT B
             WHERE B.CONTACTID = SYSDBA.TICKET.CONTACTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> 12-02: Tickets with a different Account and Contact Account ID (Fix Account with Contact Account ID)
<<',@TestCount)

-- >> 12-03-A: Tickets with no Account and Contact ID (Delete Ticket) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE (ACCOUNTID IS NULL) AND (CONTACTID IS NULL)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> 12-03-A: Tickets with no Account and Contact ID (Delete Ticket) <<',@TestCount)

-- >> 12-03-B: Tickets with invalid AccountID and invalid ContactId (Delete Ticket) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE (ACCOUNTID IS NOT NULL) AND (NOT EXISTS
        (SELECT B.ACCOUNTID FROM SYSDBA.ACCOUNT B
             WHERE B.ACCOUNTID = SYSDBA.TICKET.ACCOUNTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> 12-03-B: Tickets with invalid AccountID and invalid ContactId (Delete Ticket) <<',@TestCount)

-- >> 12-03-C: Ticket Solutions with no or invalid TicketID (Delete TicketSolution) <<
   SELECT @TestCount=COUNT(*) FROM SYSDBA.TICKETSOLUTION
   WHERE (TICKETID IS NULL) OR (NOT EXISTS
        (SELECT B.TICKETID FROM SYSDBA.TICKET B
             WHERE B.TICKETID = SYSDBA.TICKETSOLUTION.TICKETID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> 12-03-C: Ticket Solutions with no or invalid TicketID (Delete TicketSolution) <<',@TestCount)

-- >> 12-03-D: Ticket Problems with no or invalid TicketID (Delete TicketProblem) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKETPROBLEM
   WHERE (TICKETID IS NULL) OR (NOT EXISTS
        (SELECT B.TICKETID FROM SYSDBA.TICKET B
             WHERE B.TICKETID = SYSDBA.TICKETPROBLEM.TICKETID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
        Values('>> 12-03-D: Ticket Problems with no or invalid TicketID (Delete TicketProblem) <<',@TestCount)

-- >> 12-03-E: Ticket Activities with no or invalid TicketID (Delete TicketActivity) <<
   SELECT @TestCount=COUNT(TICKETACTIVITYID) FROM SYSDBA.TICKETACTIVITY
   WHERE (TICKETID IS NULL) OR (NOT EXISTS
        (SELECT B.TICKETID FROM SYSDBA.TICKET B
             WHERE B.TICKETID = SYSDBA.TICKETACTIVITY.TICKETID))

   IF @TestCount<>0
```

```
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-03-E: Ticket Activities with no or invalid TicketID (Delete TicketActivity) <<',@TestCount)

-- >> 12-03-F: Ticket Activity Items with no or invalid TicketID (Delete TicketActivityItem) <<
   SELECT @TestCount=COUNT(TICKETACTIVITYITEMID) FROM SYSDBA.TICKETACTIVITYITEM
   WHERE (TICKETACTIVITYID IS NULL) OR (NOT EXISTS
         (SELECT B.TICKETID FROM SYSDBA.TICKETACTIVITY B
               WHERE B.TICKETACTIVITYID = SYSDBA.TICKETACTIVITYITEM.TICKETACTIVITYID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-03-F: Ticket Activity Items with no or invalid TicketID (Delete TicketActivityItem) <<',@TestCount)

-- >> 12-03-G: Ticket History with no or invalid TicketID (Delete TicketHistory) <<
   SELECT @TestCount=COUNT(TICKETHISID) FROM SYSDBA.TICKETHISTORY
   WHERE (TICKETID IS NULL) OR (NOT EXISTS
         (SELECT B.TICKETID FROM SYSDBA.TICKET B
               WHERE B.TICKETID = SYSDBA.TICKETHISTORY.TICKETID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-03-G: Ticket History with no or invalid TicketID (Delete TicketHistory) <<',@TestCount)

-- >> 12-03-H: Attachments with no or invalid TicketID (Delete Attachment) <<
   SELECT @TestCount=COUNT(ATTACHID) FROM SYSDBA.ATTACHMENT
   WHERE ((TICKETID IS NOT NULL) AND (TICKETID <> '')) AND (NOT EXISTS
         (SELECT B.TICKETID FROM SYSDBA.TICKET B
               WHERE B.TICKETID = SYSDBA.ATTACHMENT.TICKETID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-03-H: Attachments with no or invalid TicketID (Delete Attachment) <<',@TestCount)

-- >> 12-05: Tickets with invalid Urgency ID (Clear Urgency) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE (URGENCYID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.URGENCYID FROM SYSDBA.URGENCY B
               WHERE B.URGENCYID = SYSDBA.TICKET.URGENCYID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-05: Tickets with invalid Urgency ID (Clear Urgency) <<',@TestCount)

-- >> 12-07: Tickets with invalid AssignedTo ID (Clear AssignedToID) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE (ASSIGNEDTOID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.SECCODEID FROM SYSDBA.SECCODE B
               WHERE B.SECCODEID = SYSDBA.TICKET.ASSIGNEDTOID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-07: Tickets with invalid AssignedTo ID (Clear AssignedToID) <<',@TestCount)

-- >> 12-08: Tickets with invalid CompletedBy ID (Clear CompletedByID) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE (COMPLETEDBYID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.SECCODEID FROM SYSDBA.SECCODE B
               WHERE B.SECCODEID = SYSDBA.TICKET.COMPLETEDBYID))
```

```
   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-08: Tickets with invalid CompletedBy ID (Clear CompletedByID) <<',@TestCount)

-- >> 12-09: Tickets with invalid ReceivedBy ID (Clear ReceivedByID) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE (RECEIVEDBYID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.SECCODEID FROM SYSDBA.SECCODE B
               WHERE B.SECCODEID = SYSDBA.TICKET.RECEIVEDBYID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-09: Tickets with invalid ReceivedBy ID (Clear ReceivedByID) <<',@TestCount)

-- >> 12-10: Tickets with no owner (Set to Account owner) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE (SECCODEID IS NULL) AND (ACCOUNTID IS NOT NULL)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-10: Tickets with no owner (Set to Account owner) <<',@TestCount)

-- >> 12-11: Tickets with invalid owner (Set to Everyone) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE (SECCODEID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.SECCODEID FROM SYSDBA.SECCODE B
               WHERE B.SECCODEID = SYSDBA.TICKET.SECCODEID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-11: Tickets with invalid owner (Set to Everyone) <<',@TestCount)

-- >> 12-13: Tickets with invalid StandardSolution ID (Clear StandardSolutionID) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE (STANDARDSOLUTIONID IS NOT NULL) AND
         (NOT EXISTS (SELECT B.TICKETSOLUTIONTYPEID FROM SYSDBA.TICKETSOLUTIONTYPE B
               WHERE B.TICKETSOLUTIONTYPEID = SYSDBA.TICKET.STANDARDSOLUTIONID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-13: Tickets with invalid StandardSolution ID (Clear StandardSolutionID) <<',@TestCount)

-- >> 12-12: Tickets with invalid StandardProblem ID (Clear StandardProblemID) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE (STANDARDPROBLEMID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.TICKETPROBLEMTYPEID FROM SYSDBA.TICKETPROBLEMTYPE B
               WHERE B.TICKETPROBLEMTYPEID = SYSDBA.TICKET.STANDARDPROBLEMID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-12: Tickets with invalid StandardProblem ID (Clear StandardProblemID) <<',@TestCount)

-- >> 12-06: Tickets with invalid Contract ID (Clear ContractID) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE (CONTRACTID IS NOT NULL) AND (NOT EXISTS
         (SELECT B.CONTRACTID FROM SYSDBA.CONTRACT B
               WHERE B.CONTRACTID = SYSDBA.TICKET.CONTRACTID))
```

```
   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-06: Tickets with invalid Contract ID (Clear ContractID) <<',@TestCount)

-- >> 12-14: Tickets with missing TicketProblem (Create TicketProblem) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE (NOT EXISTS (SELECT B.TICKETID FROM SYSDBA.TICKETPROBLEM B
                  WHERE B.TICKETID = SYSDBA.TICKET.TICKETID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-14: Tickets with missing TicketProblem (Create TicketProblem) <<',@TestCount)

-- >> 12-15: Tickets with missing TicketSolution (Create TicketSolution) <<
   SELECT @TestCount=COUNT(TICKETID) FROM SYSDBA.TICKET
   WHERE (NOT EXISTS (SELECT B.TICKETID FROM SYSDBA.TICKETSOLUTION B
                  WHERE B.TICKETID = SYSDBA.TICKET.TICKETID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-15: Tickets with missing TicketSolution (Create TicketSolution) <<',@TestCount)

-- >> 12-19: Ticket Activity Items with invalid Ticket Activity ID (Delete TicketActivityItem) <<
   SELECT @TestCount=COUNT(TICKETACTIVITYITEMID) FROM SYSDBA.TICKETACTIVITYITEM
   WHERE (TICKETACTIVITYID IS NULL) OR (NOT EXISTS
         (SELECT B.TICKETACTIVITYID FROM SYSDBA.TICKETACTIVITY B
                  WHERE B.TICKETACTIVITYID = SYSDBA.TICKETACTIVITYITEM.TICKETACTIVITYID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-19: Ticket Activity Items with invalid Ticket Activity ID (Delete TicketActivityItem) <<',@TestCount)

-- >> 12-20: Ticket Activity Items with invalid Product ID (Delete TicketActivityItem) <<
   SELECT @TestCount=COUNT(TICKETACTIVITYITEMID) FROM SYSDBA.TICKETACTIVITYITEM
   WHERE (PRODUCTID IS NULL) OR (NOT EXISTS
         (SELECT B.PRODUCTID FROM SYSDBA.PRODUCT B
                  WHERE B.PRODUCTID = SYSDBA.TICKETACTIVITYITEM.PRODUCTID))

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> 12-20: Ticket Activity Items with invalid Product ID (Delete TicketActivityItem) <<',@TestCount)

-- >> User Security with bad Manager IDs (Clear) <<
   SELECT @TestCount=COUNT(USERID) FROM SYSDBA.USERSECURITY
   WHERE (MANAGERID IS NOT NULL) AND (MANAGERID <> '') AND NOT EXISTS
         (SELECT USERID FROM SYSDBA.USERINFO A2
                  WHERE SYSDBA.USERSECURITY.MANAGERID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> User Security with bad Manager IDs (Clear) <<',@TestCount)

-- >> User Info with bad Address IDs (Create) <<
   SELECT @TestCount=COUNT(USERID) FROM SYSDBA.USERINFO A1
   WHERE NOT EXISTS (SELECT ADDRESSID FROM SYSDBA.ADDRESS A2
         WHERE A1.ADDRESSID = A2.ADDRESSID)
```

```
   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> User Info with bad Address IDs (Create) <<',@TestCount)

-- >> User Info with bad Home Address IDs (Create) <<
   SELECT @TestCount=COUNT(USERID) FROM SYSDBA.USERINFO A1
   WHERE (A1.HOMEADDRESSID <> '') AND (A1.HOMEADDRESSID IS NOT NULL) AND NOT EXISTS
         (SELECT ADDRESSID FROM SYSDBA.ADDRESS A2 WHERE A1.HOMEADDRESSID = A2.ADDRESSID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> User Info with bad Home Address IDs (Create) <<',@TestCount)

-- >> User View records not tied to a User (Delete) <<
   SELECT @TestCount=COUNT(USERVIEWID) FROM SYSDBA.USERVIEW
   WHERE NOT EXISTS (SELECT USERID FROM SYSDBA.USERINFO A2
               WHERE SYSDBA.USERVIEW.USERID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> User View records not tied to a User (Delete) <<',@TestCount)

-- >> User Option records not tied to a User (Delete) <<
   SELECT @TestCount=COUNT(OPTIONID) FROM SYSDBA.USEROPTIONS
   WHERE (USERID <> 'PROCESS') AND NOT EXISTS
         (SELECT USERID FROM SYSDBA.USERINFO A2
               WHERE SYSDBA.USEROPTIONS.USERID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> User Option records not tied to a User (Delete) <<',@TestCount)

-- >> User Security records not in User Info (Delete) <<
   SELECT @TestCount=COUNT(USERID) FROM SYSDBA.USERSECURITY
   WHERE NOT EXISTS (SELECT USERID FROM SYSDBA.USERINFO A2
         WHERE SYSDBA.USERSECURITY.USERID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> User Security records not in User Info (Delete) <<',@TestCount)

-- >> User Info records not in User Security (Delete) <<
   SELECT @TestCount=COUNT(USERID) FROM SYSDBA.USERINFO
   WHERE NOT EXISTS (SELECT USERID FROM SYSDBA.USERSECURITY A2
         WHERE SYSDBA.USERINFO.USERID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> User Info records not in User Security (Delete) <<',@TestCount)

-- >> User Profile records not in User Security (Delete) <<
   SELECT @TestCount=COUNT(USERID) FROM SYSDBA.USERPROFILE
   WHERE NOT EXISTS (SELECT USERID FROM SYSDBA.USERSECURITY A2
         WHERE SYSDBA.USERPROFILE.USERID = A2.USERID)

   IF @TestCount<>0
   Insert into LCMS_IntegrityChecks (TestName,TestCount)
         Values('>> User Profile records not in User Security (Delete) <<',@TestCount)
```

GO

sp_LCMS_IntegrityCheck_SendEmail (Stored Procedure)

```
CREATE PROCEDURE [dbo].[sp_LCMS_IntegrityCheck_SendEmail] AS
declare @ErrorCount as int
declare @eSubject as varchar(150)
declare @eBody as varchar(8000)
declare @TestName as varchar(150)
declare @TestCount as varchar(10)
declare @LoopCount as int


select @ErrorCount=count(0)
from lcms_IntegrityChecks

set @eSubject=cast(@ErrorCount as varchar(3)) + ' Integrity Check(s) Failed on ' + cast(getdate() as varchar(25))

declare curErrors Cursor For
select TestName,cast(TestCount as varchar(10)) as TestCount from lcms_IntegrityChecks

Open curErrors

Fetch Next From curErrors Into @TestName,@TestCount
--select @TestName,@TestCount
set @LoopCount=1

While @@Fetch_Status = 0 and @LoopCount < 11    --Only show 1st 10 records
Begin
        set @eBody=isnull(@eBody,'') + @TestCount + ' tests failed for ' + @TestName + char(13) + char(10)
        Fetch Next From curErrors Into @TestName,@TestCount
        set @LoopCount=@LoopCount+1
End
select char(13),char(10)

Close curErrors
Deallocate curErrors

select @eBody

if @ErrorCount>0
begin
   exec sp_send_cdosysmail
        @From='SalesLogix.Administrator@lcms.org',
        @To='dave.mcgill@concordiatech.org;Murel.Warren@ConcordiaTech.org;Pat.Ulmer@ConcordiaTech.org',
        @Subject=@eSubject,
        @Body=@eBody
end
GO
```

Tables

Automated SalesLogix Integrity Checks