Acmeware

Data Repository Experts Since 1998

Soup to Nuts – Data Repository 102

2018 MUSE International

Presenter: Jamie Gerardo

Today’s Agenda

- Finding Data
- Report Development Standards
- Writing Efficient Code
- T-SQL Tips
- Report Development

Acmeware

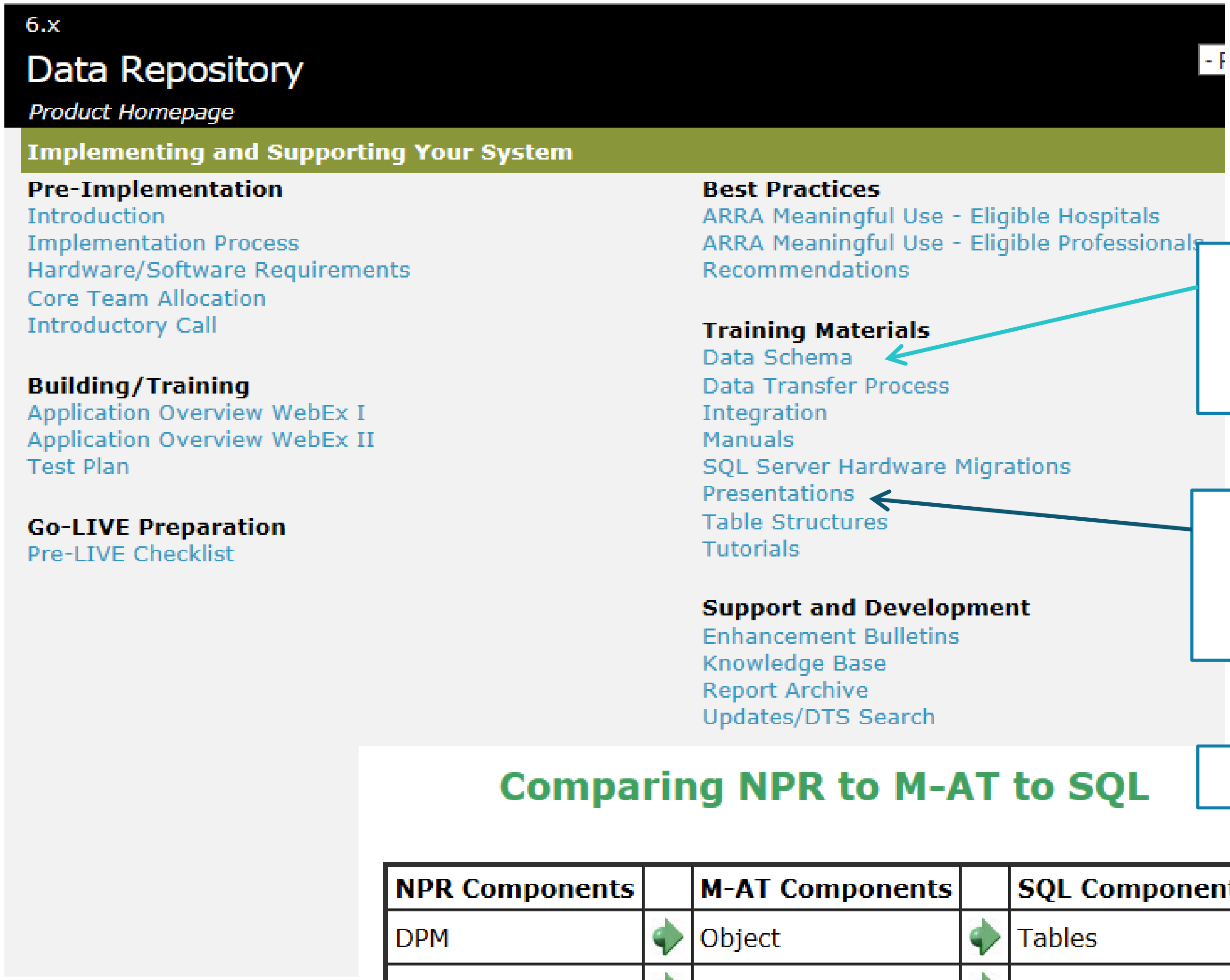
Finding Data

- If you know the NPR structure then finding data will be much easier
- In general you can think of a Detail Segment as a table

Tools

1. Meditech Website
2. SysDrTables/SysDrColumns
3. Shift F9 and Shift F8 for Magic
4. DR Application Menu

Meditech DR Home Page



6.x

Data Repository

Product Homepage

Implementing and Supporting Your System

Pre-Implementation
Introduction
Implementation Process
Hardware/Software Requirements
Core Team Allocation
Introductory Call

Building/Training
Application Overview WebEx I
Application Overview WebEx II
Test Plan

Go-LIVE Preparation
Pre-LIVE Checklist

Best Practices
ARRA Meaningful Use - Eligible Hospitals
ARRA Meaningful Use - Eligible Professionals
Recommendations

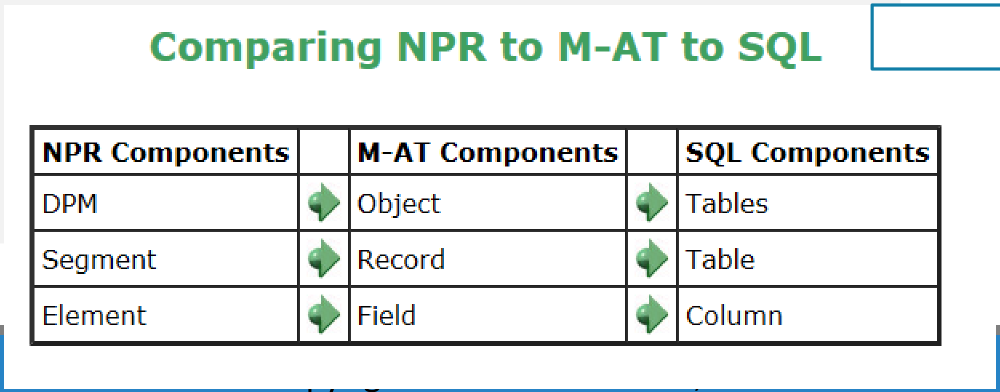
Training Materials
Data Schema
Data Transfer Process
Integration
Manuals
SQL Server Hardware Migrations
Presentations
Table Structures
Tutorials

Support and Development
Enhancement Bulletins
Knowledge Base
Report Archive
Updates/DTS Search

* Table Structure

Overview of the data repository

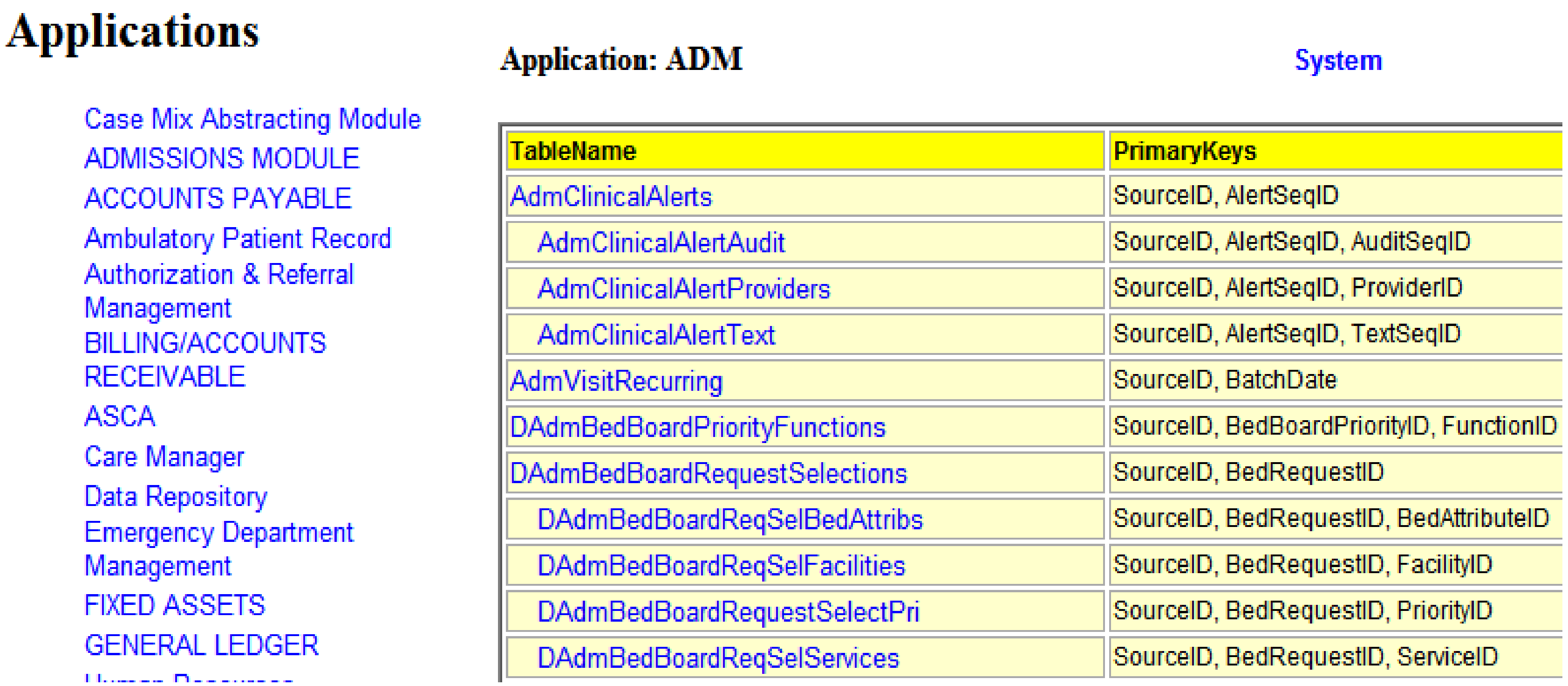
Meditech Website



Comparing NPR to M-AT to SQL

NPR Components		M-AT Components		SQL Components
DPM	➡	Object	➡	Tables
Segment	➡	Record	➡	Table
Element	➡	Field	➡	Column

Meditech – Data Model



Applications

- Case Mix Abstracting Module
- ADMISSIONS MODULE
- ACCOUNTS PAYABLE
- Ambulatory Patient Record
- Authorization & Referral Management
- BILLING/ACCOUNTS RECEIVABLE
- ASCA
- Care Manager
- Data Repository
- Emergency Department Management
- FIXED ASSETS
- GENERAL LEDGER

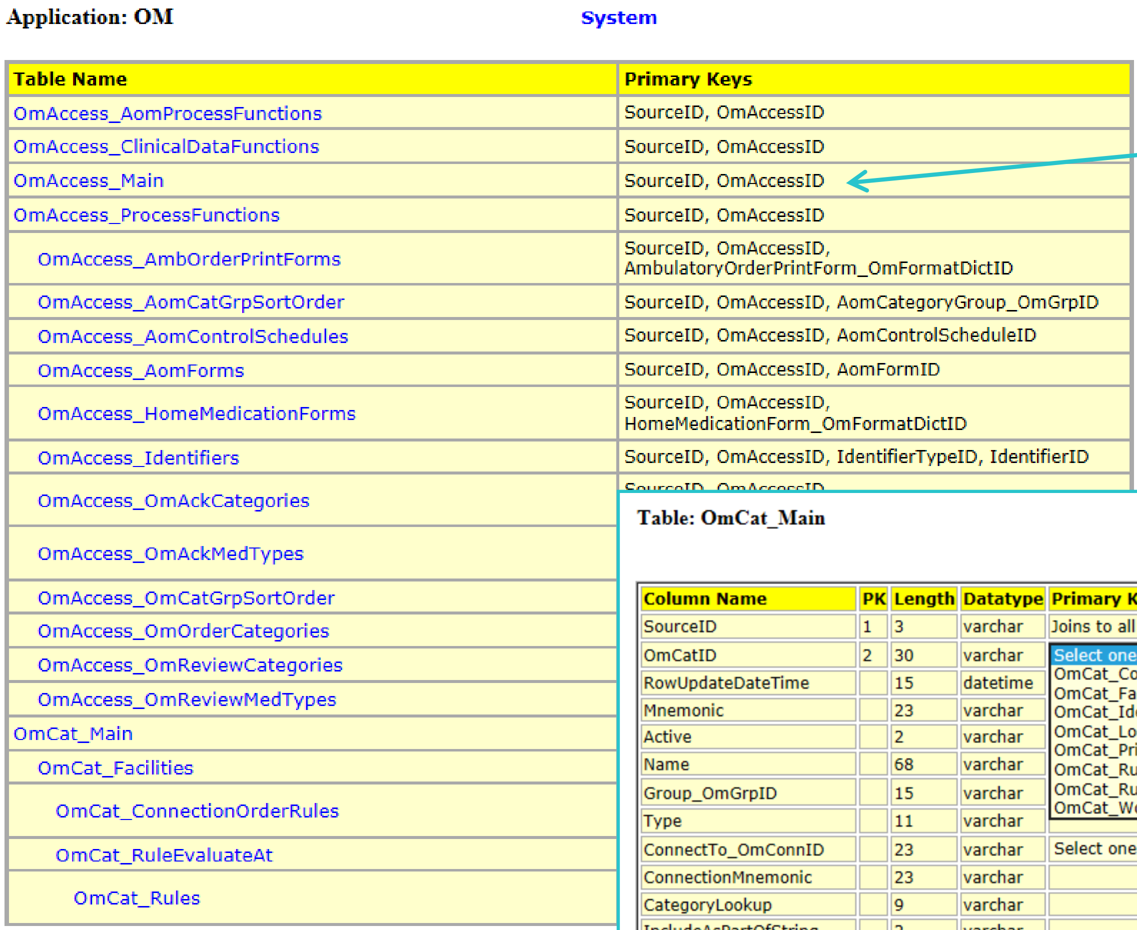
Application: ADM

System

TableName	PrimaryKeys
AdmClinicalAlerts	SourceID, AlertSeqID
AdmClinicalAlertAudit	SourceID, AlertSeqID, AuditSeqID
AdmClinicalAlertProviders	SourceID, AlertSeqID, ProviderID
AdmClinicalAlertText	SourceID, AlertSeqID, TextSeqID
AdmVisitRecurring	SourceID, BatchDate
DAdmBedBoardPriorityFunctions	SourceID, BedBoardPriorityID, FunctionID
DAdmBedBoardRequestSelections	SourceID, BedRequestID
DAdmBedBoardReqSelBedAttribs	SourceID, BedRequestID, BedAttributeID
DAdmBedBoardReqSelFacilities	SourceID, BedRequestID, FacilityID
DAdmBedBoardRequestSelectPri	SourceID, BedRequestID, PriorityID
DAdmBedBoardReqSelServices	SourceID, BedRequestID, ServiceID

1. Shows the equivalent NPR – Parent/Child relationships
2. Interactive Primary Keys that displays other tables with foreign keys

Meditech 6.0 – Data Model



Application: OM System

Table Name	Primary Keys
OmAccess_AomProcessFunctions	SourceID, OmAccessID
OmAccess_ClinicalDataFunctions	SourceID, OmAccessID
OmAccess_Main	SourceID, OmAccessID
OmAccess_ProcessFunctions	SourceID, OmAccessID
OmAccess_AmbOrderPrintForms	SourceID, OmAccessID, AmbulatoryOrderPrintForm_OmFormatDictID
OmAccess_AomCatGrpSortOrder	SourceID, OmAccessID, AomCategoryGroup_OmGrpID
OmAccess_AomControlSchedules	SourceID, OmAccessID, AomControlScheduleID
OmAccess_AomForms	SourceID, OmAccessID, AomFormID
OmAccess_HomeMedicationForms	SourceID, OmAccessID, HomeMedicationForm_OmFormatDictID
OmAccess_Identifiers	SourceID, OmAccessID, IdentifierTypeID, IdentifierID
OmAccess_OmAckCategories	SourceID, OmAccessID
OmAccess_OmAckMedTypes	
OmAccess_OmCatGrpSortOrder	
OmAccess_OmOrderCategories	
OmAccess_OmReviewCategories	
OmAccess_OmReviewMedTypes	
OmCat_Main	
OmCat_Facilities	
OmCat_ConnectionOrderRules	
OmCat_RuleEvaluateAt	
OmCat_Rules	

Primary key and foreign keys

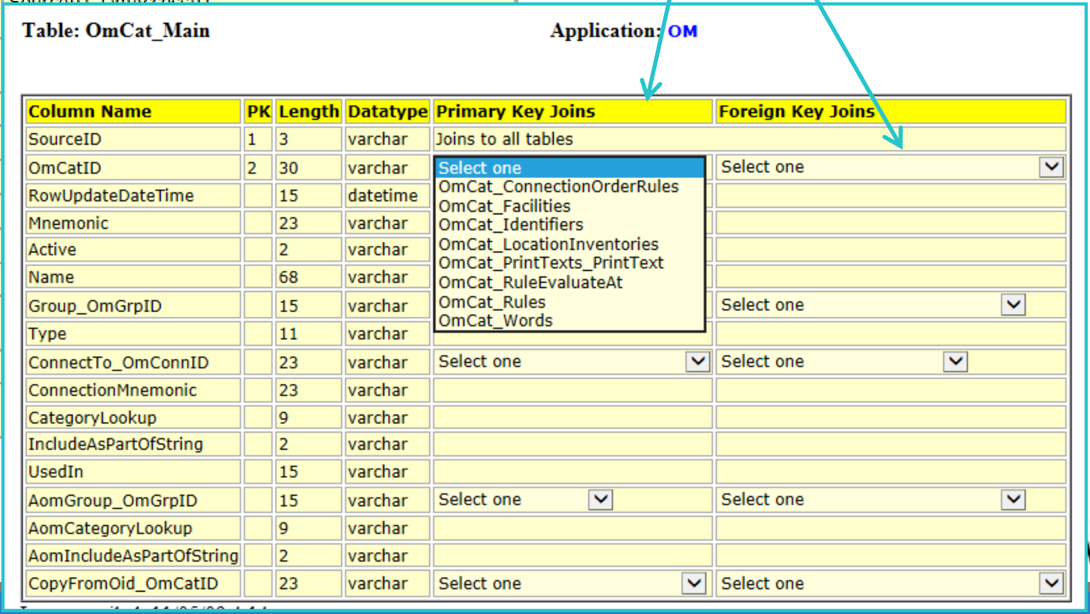


Table: OmCat_Main **Application:** OM

Column Name	PK	Length	Datatype	Primary Key Joins	Foreign Key Joins
SourceID	1	3	varchar	Joins to all tables	
OmCatID	2	30	varchar	Select one OmCat_ConnectionOrderRules OmCat_Facilities OmCat_Identifiers OmCat_LocationInventories OmCat_PrintTexts_PrintText OmCat_RuleEvaluateAt OmCat_Rules OmCat_Words	Select one
RowUpdateDateTime		15	datetime		
Mnemonic		23	varchar		
Active		2	varchar		
Name		68	varchar		
Group_OmGrpID		15	varchar		Select one
Type		11	varchar		
ConnectTo_OmConnID		23	varchar	Select one	Select one
ConnectionMnemonic		23	varchar		
CategoryLookup		9	varchar		
IncludeAsPartOfString		2	varchar		
UsedIn		15	varchar		
AomGroup_OmGrpID		15	varchar	Select one	Select one
AomCategoryLookup		9	varchar		
AomIncludeAsPartOfString		2	varchar		
CopyFromOid_OmCatID		23	varchar	Select one	Select one

Table Information in livedb and livefdb

livedb

```
---------------------------------------------------------------------------
-- A general search by DR Field Name
-- '%Comment%' is a wildcard search for any field with Comment
-- You can modify the name as needed for your search
---------------------------------------------------------------------------

SELECT T.Name, C.*
FROM livedb.dbo.SysDrColumns C
INNER JOIN livedb.dbo.SysDrTables T
    ON C.TableID = T.TableID
WHERE C.Name like '%Comment%’
ORDER BY 1

---------------------------------------------
-- A search by specific NPR field
---------------------------------------------

SELECT T.Name, C.*
FROM livedb.dbo.SysDrColumns C
INNER JOIN livedb.dbo.SysDrTables T
   ON C.TableID = T.TableID
WHERE C.NprElement = 'BAR.PAT.account'
order by 1
```

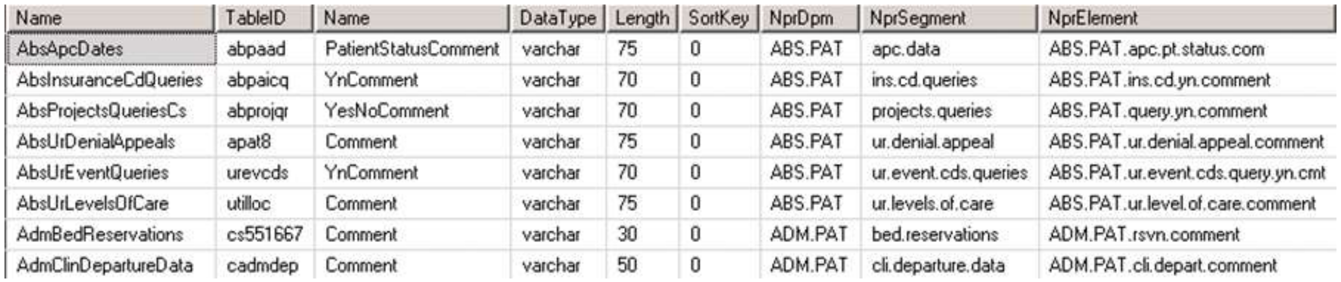
livefdb

```
---------------------------------------------------------------------------
-- A general search by DR Field Name
-- '%Comment%' is a wildcard search for any field with Comment
-- You can modify the name as needed for your search
---------------------------------------------------------------------------

SELECT DT_M.TableName, DT_C.*
FROM livefdb.dbo.DrTable_Main DT_M
INNER JOIN livefdb.dbo.DrTable_Columns DT_C
    ON DT_M.SourceID = DT_C.SourceID
AND DT_M.DrTableID = DT_C.DrTableID
WHERE DT_C.ColumnName like '%Comment%'
ORDER BY 1
---------------------------------------------
-- A search by specific NPR field
---------------------------------------------

SELECT DT_M.TableName, DT_C.*
FROM livefdb.dbo.DrTable_Main DT_M
INNER JOIN livefdb.dbo.DrTable_Columns DT_C
    ON DT_M.SourceID = DT_C.SourceID
    AND DT_M.DrTableID = DT_C.DrTableID
WHERE DT_C.ColumnObjectClass = 'OmOrd'
ORDER BY 1
```

Examples

Shows table name, column, data type
along with the DPM, NprSegment and NprElement

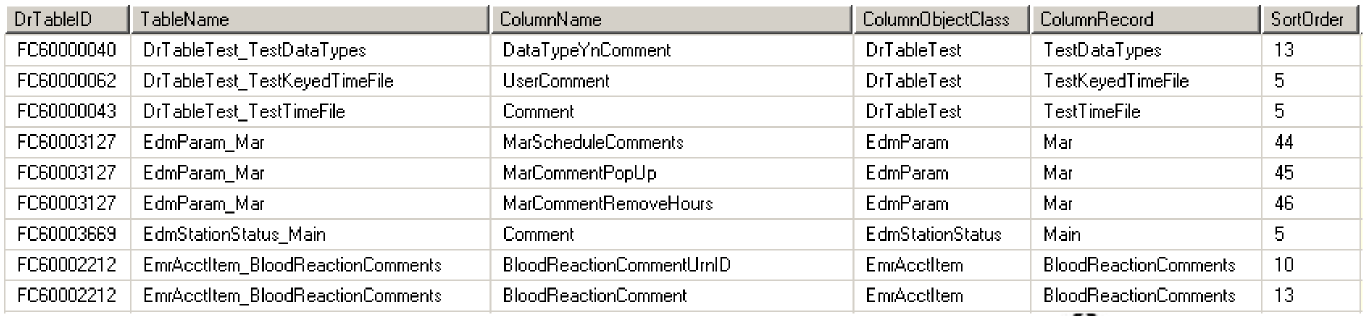
```
SELECT T.Name, C.*
FROM livedb.dbo.SysDrColumns C
INNER JOIN livedb.dbo.SysDrTables T
ON C.TableID = T.TableID
WHERE C.Name like '%Comment%'
order by 1
```



Name	TableID	Name	DataType	Length	SortKey	NprDpm	NprSegment	NprElement
AbsApcDates	abpaad	PatientStatusComment	varchar	75	0	ABS.PAT	apc.data	ABS.PAT.apc.pt.status.com
AbsInsuranceCdQueries	abpaicq	YnComment	varchar	70	0	ABS.PAT	ins.cd.queries	ABS.PAT.ins.cd.yn.comment
AbsProjectsQueriesCs	abprojqr	YesNoComment	varchar	70	0	ABS.PAT	projects.queries	ABS.PAT.query.yn.comment
AbsUrDenialAppeals	apat8	Comment	varchar	75	0	ABS.PAT	ur.denial.appeal	ABS.PAT.ur.denial.appeal.comment
AbsUrEventQueries	urevcds	YnComment	varchar	70	0	ABS.PAT	ur.event.cds.queries	ABS.PAT.ur.event.cds.query.yn.cmt
AbsUrLevelsOfCare	utilloc	Comment	varchar	75	0	ABS.PAT	ur.levels.of.care	ABS.PAT.ur.level.of.care.comment
AdmBedReservations	cs551667	Comment	varchar	30	0	ADM.PAT	bed.reservations	ADM.PAT.rsvn.comment
AdmClinDepartureData	cadmdep	Comment	varchar	50	0	ADM.PAT	cli.departure.data	ADM.PAT.cli.depart.comment

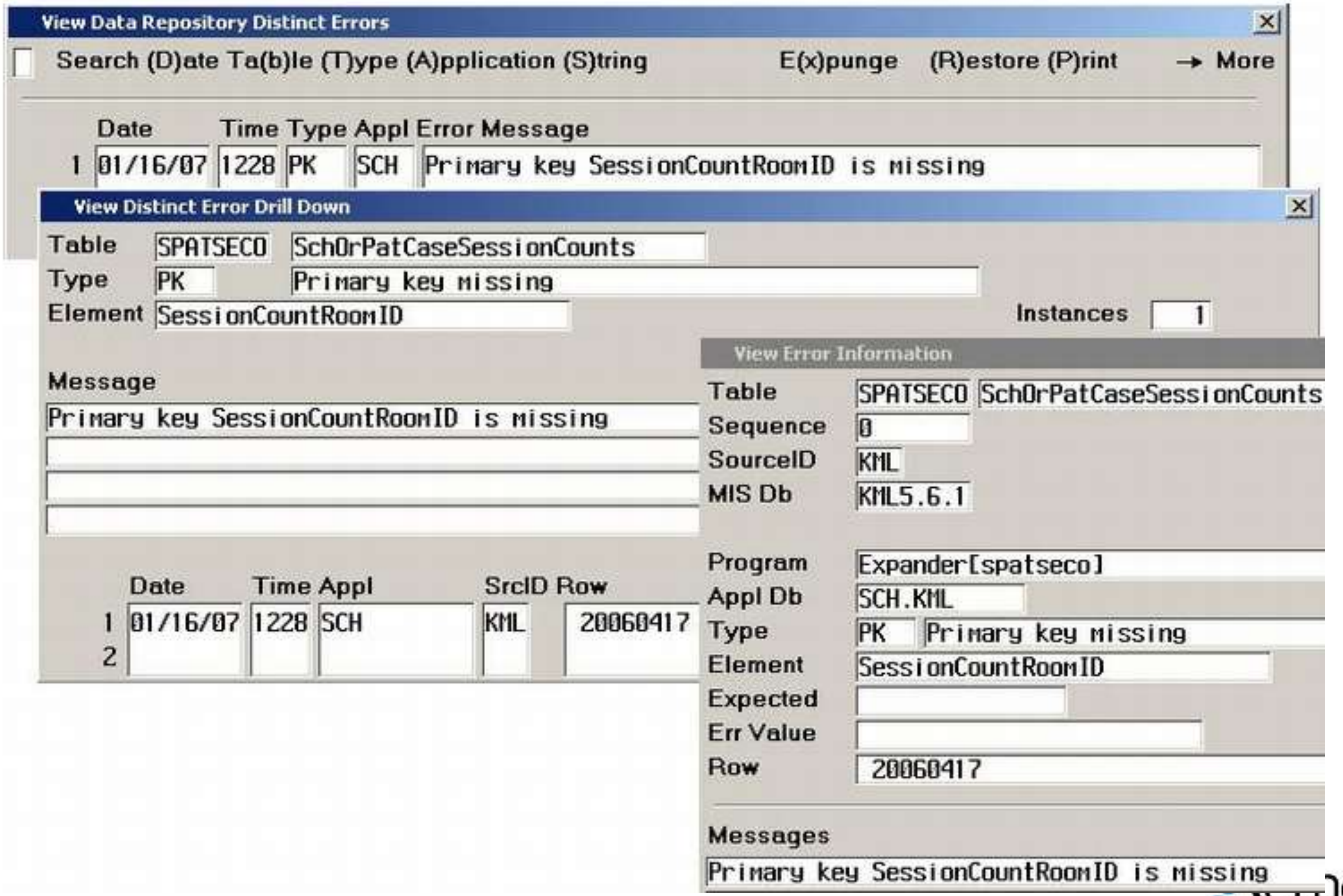
```
SELECT DT_M.TableName, DT_C.*
FROM livefoc.dbo.DrTable_Main DT_M
INNER JOIN livefoc.dbo.DrTable_Columns DT_C
ON DT_M.SourceID = DT_C.SourceID
AND DT_M.DrTableID = DT_C.DrTableID
WHERE DT_C.ColumnName  like '%Comment%'
ORDER BY 1
```

Shows table name, column, data type, length
along with ObjectClass, Column Record and Column Field



DrTableID	TableName	ColumnName	ColumnObjectClass	ColumnRecord	SortOrder
FC60000040	DrTableTest_TestDataTypes	DataTypeYnComment	DrTableTest	TestDataTypes	13
FC60000062	DrTableTest_TestKeyedTimeFile	UserComment	DrTableTest	TestKeyedTimeFile	5
FC60000043	DrTableTest_TestTimeFile	Comment	DrTableTest	TestTimeFile	5
FC60003127	EdmParam_Mar	MarScheduleComments	EdmParam	Mar	44
FC60003127	EdmParam_Mar	MarCommentPopUp	EdmParam	Mar	45
FC60003127	EdmParam_Mar	MarCommentRemoveHours	EdmParam	Mar	46
FC60003669	EdmStationStatus_Main	Comment	EdmStationStatus	Main	5
FC60002212	EmrAcctItem_BloodReactionComments	BloodReactionCommentUrnID	EmrAcctItem	BloodReactionComments	10
FC60002212	EmrAcctItem_BloodReactionComments	BloodReactionComment	EmrAcctItem	BloodReactionComments	13

Shift F9 and Shift F8 for Magic



View Data Repository Distinct Errors

Search (D)ate Ta(b)le (T)ype (A)pplication (S)tring E(x)punge (R)estore (P)rint → More

	Date	Time	Type	Appl	Error Message
1	01/16/07	1228	PK	SCH	Primary key SessionCountRoomID is missing

View Distinct Error Drill Down

Table	SPATSECO	SchOrPatCaseSessionCounts
Type	PK	Primary key missing
Element	SessionCountRoomID	

Instances 1

Message

Primary key SessionCountRoomID is missing

	Date	Time	Appl	SrcID	Row
1	01/16/07	1228	SCH	KML	20060417
2					

View Error Information

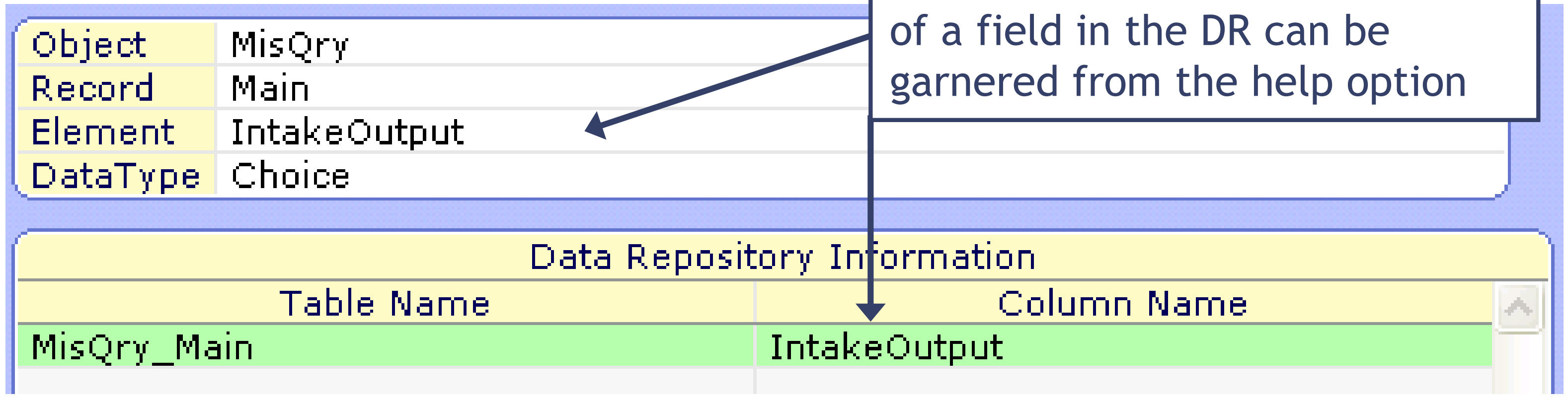
Table	SPATSECO	SchOrPatCaseSessionCounts
Sequence	0	
SourceID	KML	
MIS Db	KML5.6.1	
Program	Expander[spatseco]	
Appl Db	SCH.KML	
Type	PK	Primary key missing
Element	SessionCountRoomID	
Expected		
Err Value		
Row	20060417	

Messages

Primary key SessionCountRoomID is missing

Identifying Data Fields in the 6.0 DR

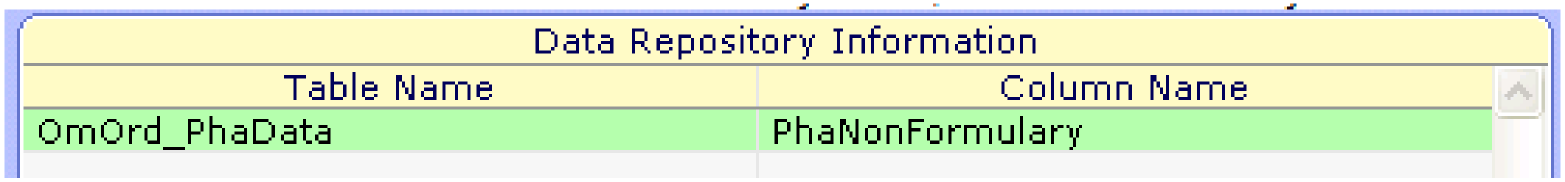
Likewise, in AT applications, information about the location of a field in the DR can be garnered from the help option



Object	MisQry
Record	Main
Element	IntakeOutput
DataType	Choice

Data Repository Information

Table Name	Column Name
MisQry_Main	IntakeOutput



Data Repository Information

Table Name	Column Name
OmOrd_PhaData	PhaNonFormulary

Table and Field Inquiry on the DR Application Menu

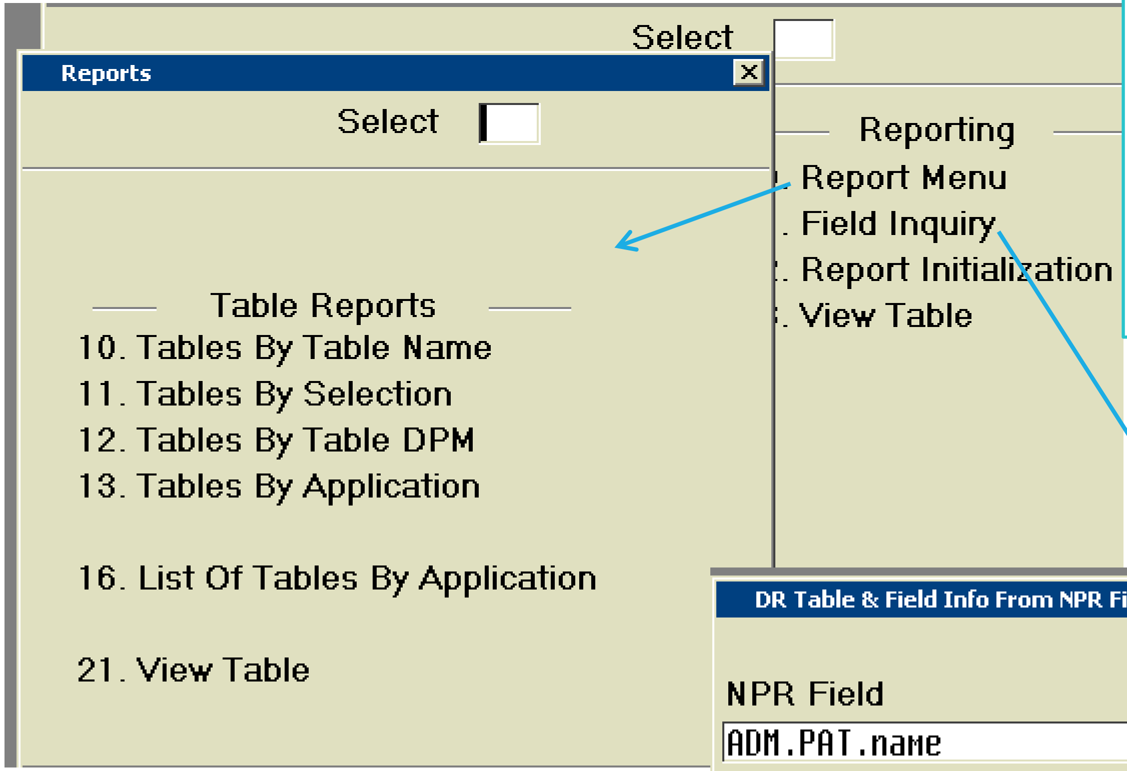


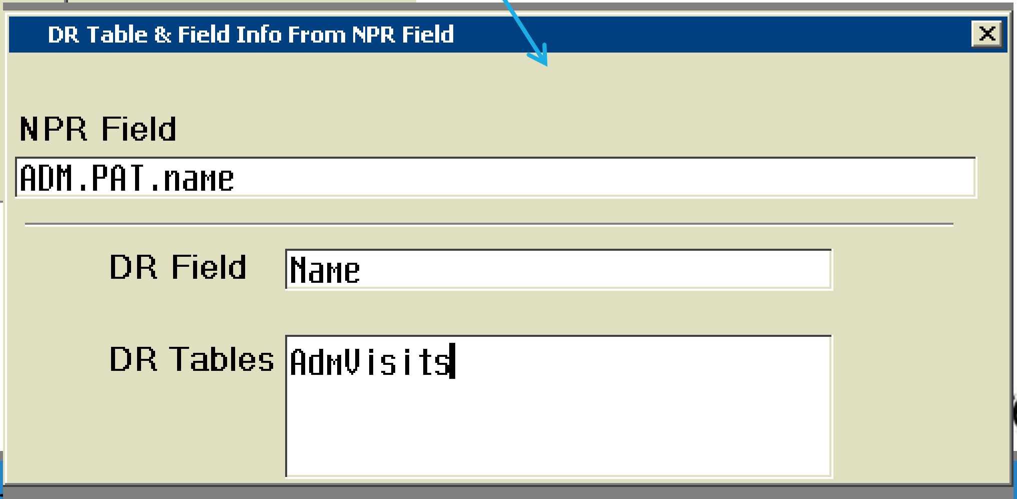
Table Reports

10. Tables By Table Name
11. Tables By Selection
12. Tables By Table DPM
13. Tables By Application

16. List Of Tables By Application

21. View Table

I rarely use these tools on the DR Menu – They can be helpful but I find the previously covered options to be the best.



DR Table & Field Info From NPR Field

NPR Field: ADM.PAT.name

DR Field: Name

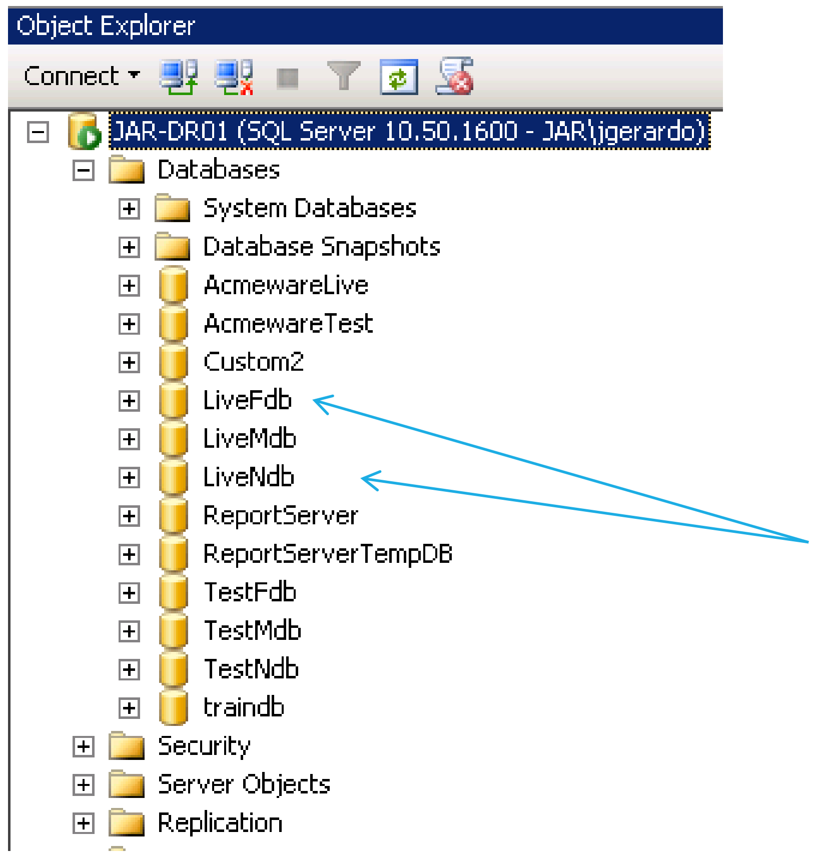
DR Tables: AdmVisits

Magic 5.66 Screen

Report/SQL Development Standards

- Stored procedures
- Data and Database Organization
- Documentation

Organization - Database


Object Explorer

Connect

```
JAR-DR01 (SQL Server 10.50.1600 - JAR\jgerardo)
  Databases
    System Databases
    Database Snapshots
    AcmewareLive
    AcmewareTest
    Custom2
    LiveFdb
    LiveMdb
    LiveNdb
    ReportServer
    ReportServerTempDB
    TestFdb
    TestMdb
    TestNdb
    traindb
  Security
  Server Objects
  Replication
```

1. Don't save stored procedures and/or tables in live databases.
2. Create a database to keep your stored procedures, views and tables.
   - Recovery Mode is set to simple
   - You can set up the database files similar to livendb
3. You will want to include the database you create in your backup plan.

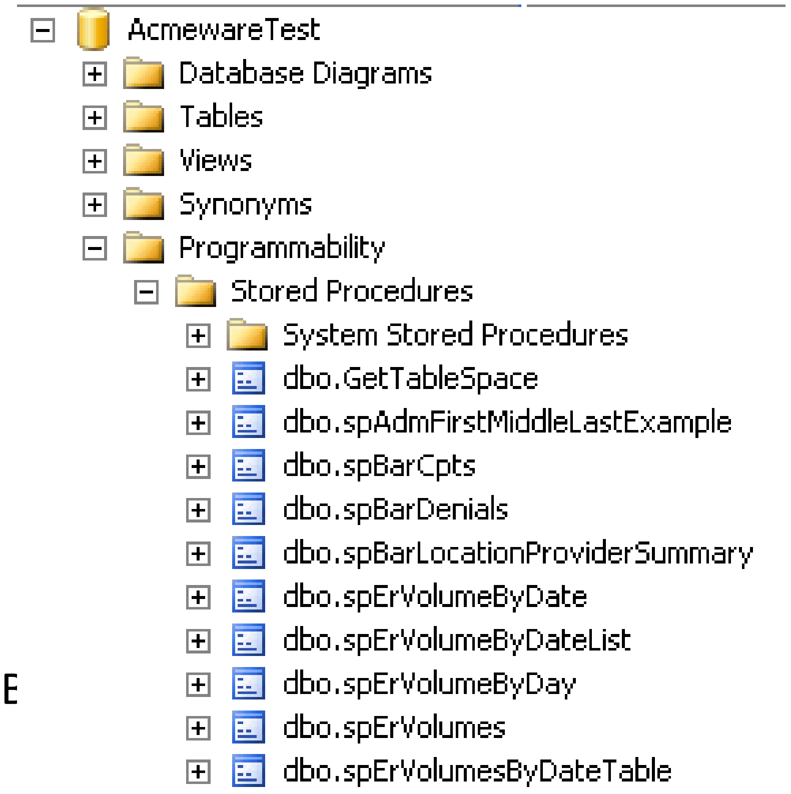
What is a stored procedure?

A stored procedure is a saved set of code on the SQL Server that allows you to run:

```
EXEC spBarAccountsByAccountType
```

Rather than……

```
SELECT
        BVFD.AccountType,
        BV.PostalCode,
        BV.AccountNumber

FROM livedb.dbo.BarVisits BV

INNER JOIN livedb.dbo.BarVisitFinancialData B
        ON BV.SourceID = BVFD.SourceID
        AND BV.BillingID = BVFD.BillingID
```



```
⊟ AcmewareTest
    ⊞ Database Diagrams
    ⊞ Tables
    ⊞ Views
    ⊞ Synonyms
    ⊟ Programmability
        ⊟ Stored Procedures
            ⊞ System Stored Procedures
            ⊞ dbo.GetTableSpace
            ⊞ dbo.spAdmFirstMiddleLastExample
            ⊞ dbo.spBarCpts
            ⊞ dbo.spBarDenials
            ⊞ dbo.spBarLocationProviderSummary
            ⊞ dbo.spErVolumeByDate
            ⊞ dbo.spErVolumeByDateList
            ⊞ dbo.spErVolumeByDay
            ⊞ dbo.spErVolumes
            ⊞ dbo.spErVolumesByDateTable
```

Organize your Stored Procedures

- Name your stored procedures so that you can easily locate them.
  - Ex: spErDepartVolumesByHour
  - Ex: spAdmRevisitsTable
- Re-name stored procs no longer in use.
  - Ex: x_spErDepartVolumesByHour
- Use Header information to describe a stored procedure's use.
- Only save useable code as a stored procedure.
- Save test code and research code as a text file or label appropriately.

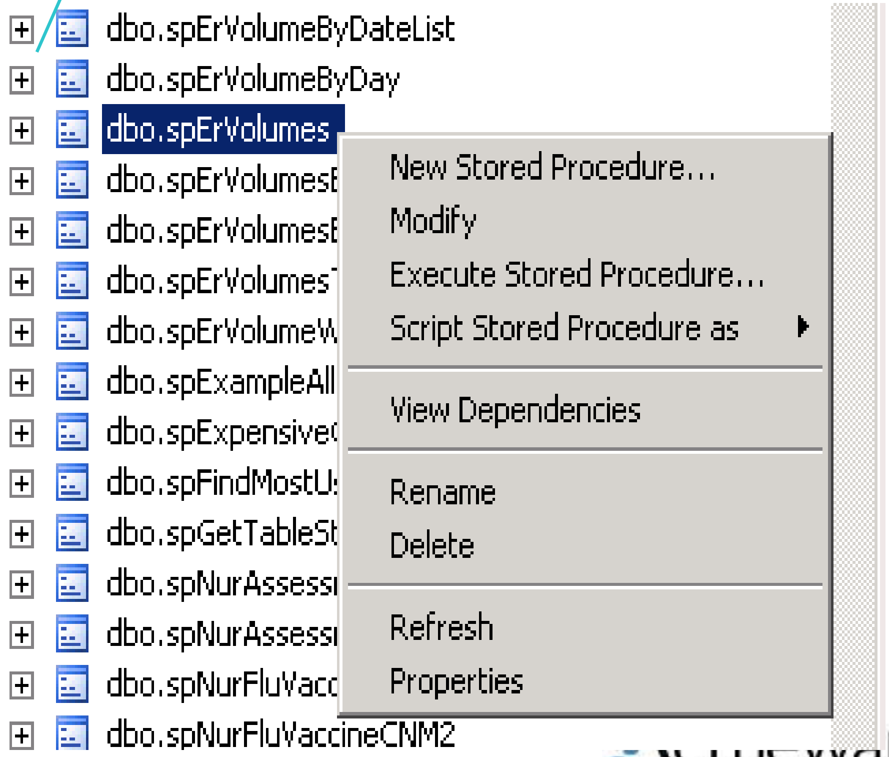
Acmeware

Stored Procedures

To save a stored procedure you CREATE PROCEDURE. This saves the stored procedure on the server/database you've selected

Once created – you can Modify and View Dependencies

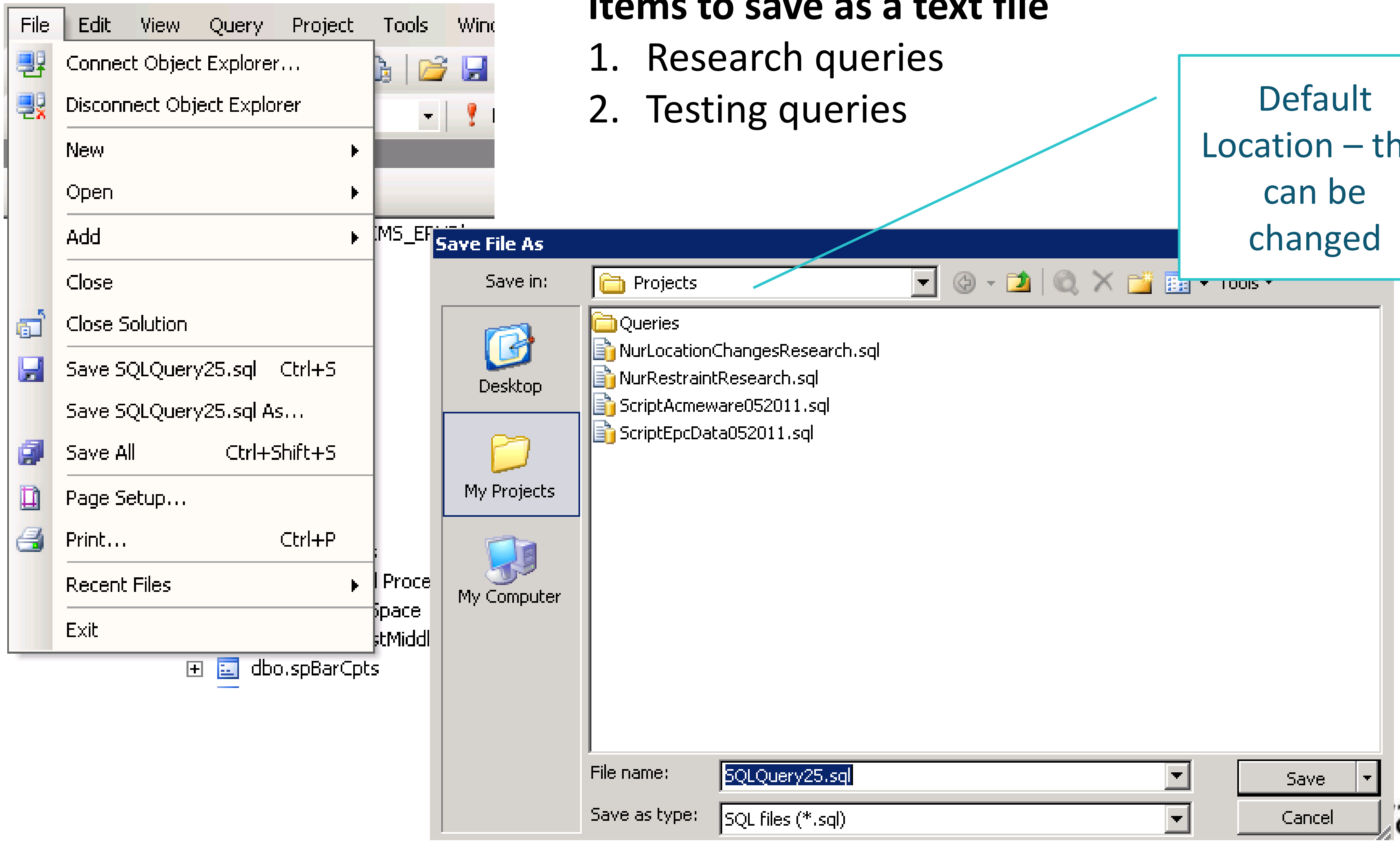
```
CREATE PROC [dbo].[spErVolumes]
(@Begin datetime,@End datetime)

AS

SELECT *
FROM tbErVolumeResults
WHERE Day_Date between @Begin and CONVERT(DA
ORDER BY 1,2
```



Saving code as a text file

Items to save as a text file

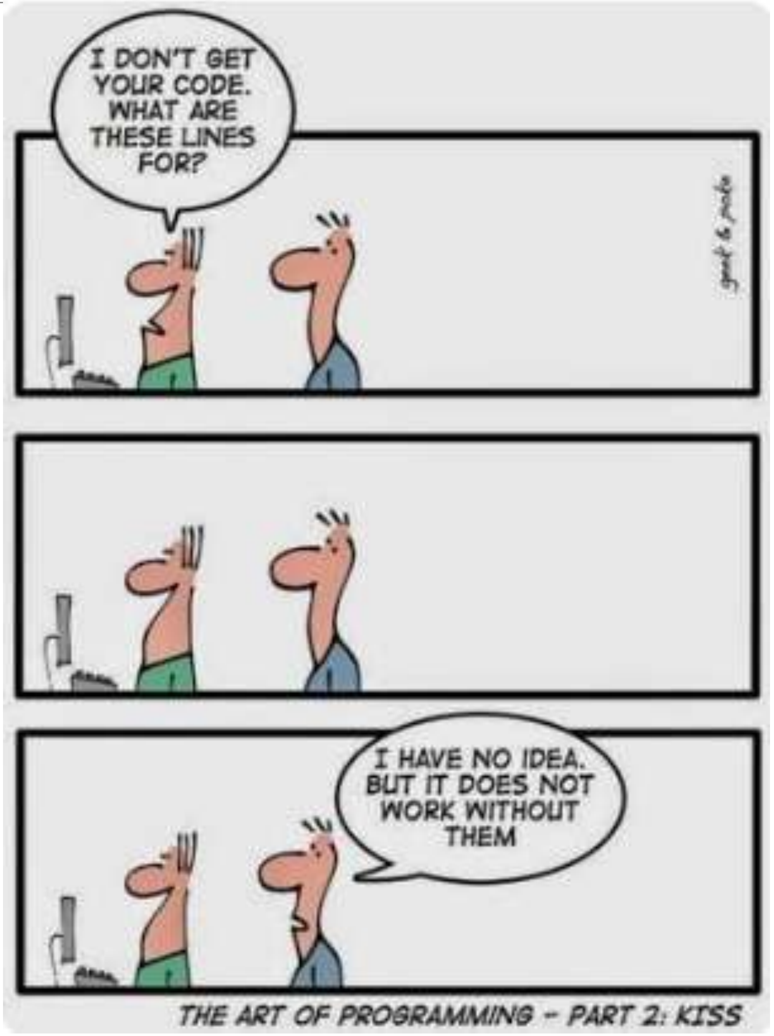
1. Research queries
2. Testing queries



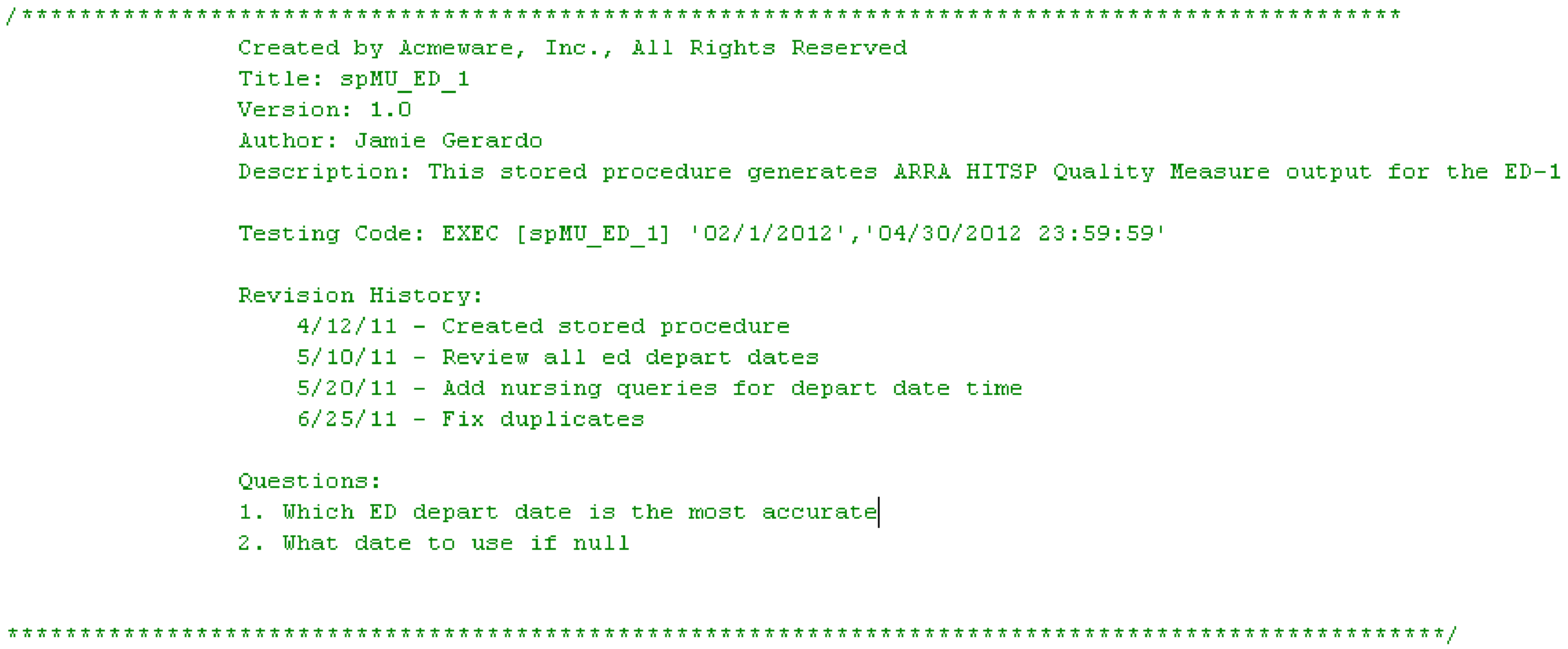
Default Location – this can be changed

Documentation

- Documenting through out your code!
- Document on any piece of code that is the least bit out of the ordinary. Not only what by why.
- Notate in each step of your code what you are doing.

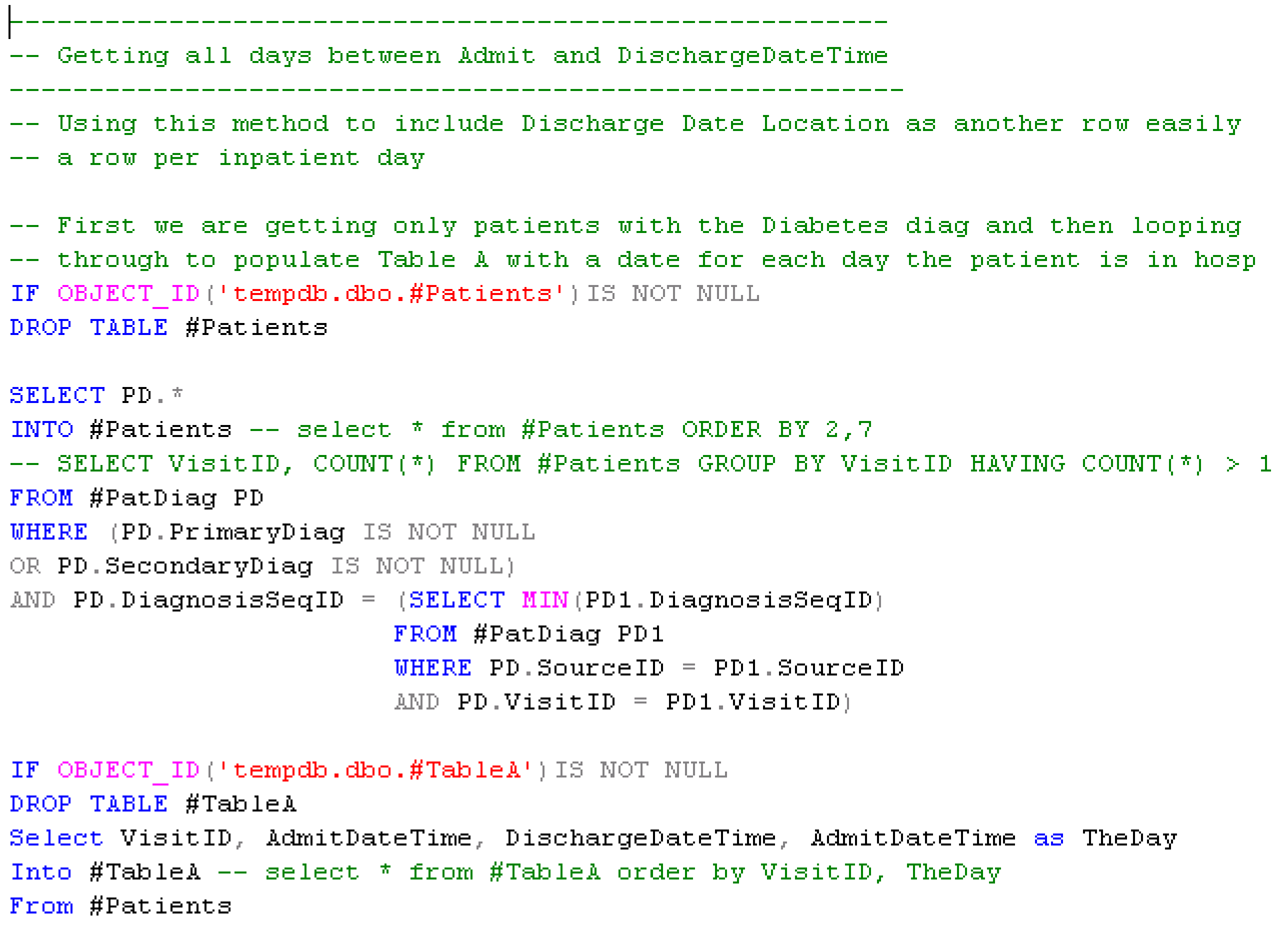

I DON'T GET YOUR CODE. WHAT ARE THESE LINES FOR?
I HAVE NO IDEA. BUT IT DOES NOT WORK WITHOUT THEM
THE ART OF PROGRAMMING - PART 2: KISS

Stored Procedure – Example Header



```
/***********************************************************************************************
                Created by Acmeware, Inc., All Rights Reserved
                Title: spMU_ED_1
                Version: 1.0
                Author: Jamie Gerardo
                Description: This stored procedure generates ARRA HITSP Quality Measure output for the ED-1

                Testing Code: EXEC [spMU_ED_1] '02/1/2012','04/30/2012 23:59:59'

                Revision History:
                    4/12/11 - Created stored procedure
                    5/10/11 - Review all ed depart dates
                    5/20/11 - Add nursing queries for depart date time
                    6/25/11 - Fix duplicates

                Questions:
                1. Which ED depart date is the most accurate
                2. What date to use if null


***************************************************************************************************/
```

Example code documentation

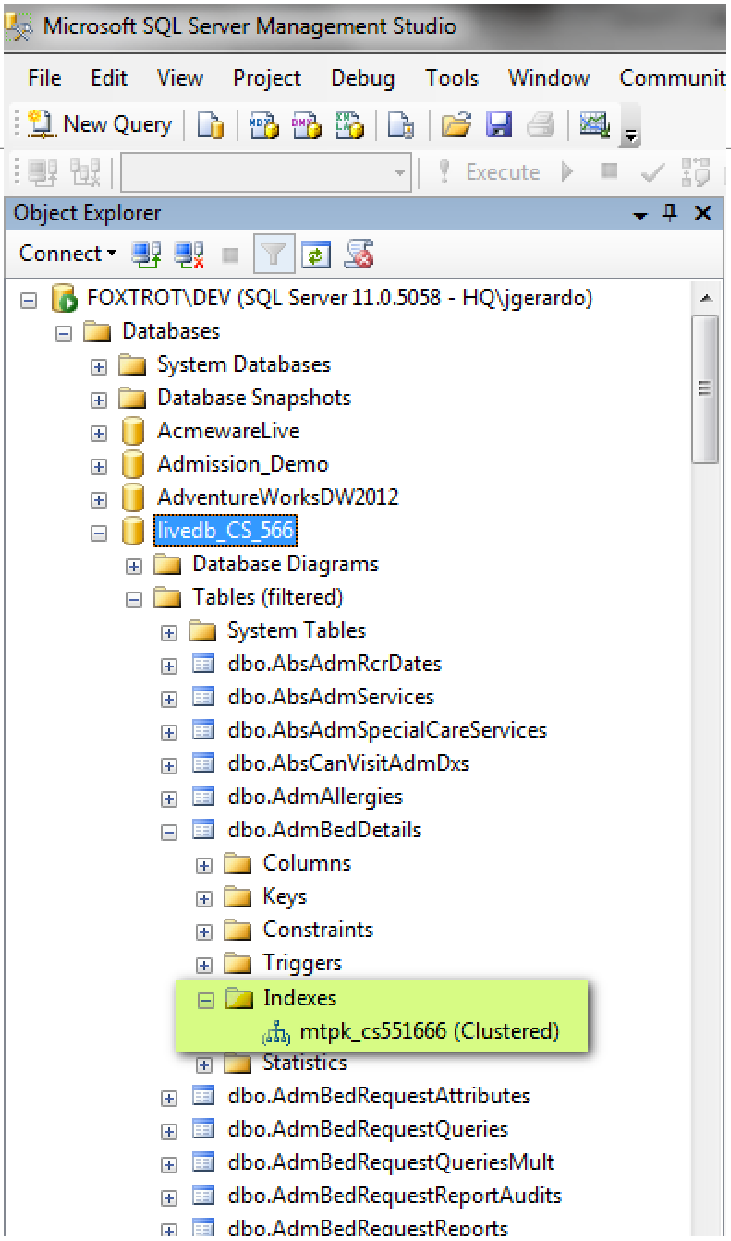


```
------------------------------------------------------
-- Getting all days between Admit and DischargeDateTime
--------------------------------------------------------
-- Using this method to include Discharge Date Location as another row easily
-- a row per inpatient day

-- First we are getting only patients with the Diabetes diag and then looping
-- through to populate Table A with a date for each day the patient is in hosp
IF OBJECT_ID('tempdb.dbo.#Patients') IS NOT NULL
DROP TABLE #Patients

SELECT PD.*
INTO #Patients -- select * from #Patients ORDER BY 2,7
-- SELECT VisitID, COUNT(*) FROM #Patients GROUP BY VisitID HAVING COUNT(*) > 1
FROM #PatDiag PD
WHERE (PD.PrimaryDiag IS NOT NULL
OR PD.SecondaryDiag IS NOT NULL)
AND PD.DiagnosisSeqID = (SELECT MIN(PD1.DiagnosisSeqID)
                        FROM #PatDiag PD1
                        WHERE PD.SourceID = PD1.SourceID
                        AND PD.VisitID = PD1.VisitID)

IF OBJECT_ID('tempdb.dbo.#TableA') IS NOT NULL
DROP TABLE #TableA
Select VisitID, AdmitDateTime, DischargeDateTime, AdmitDateTime as TheDay
Into #TableA -- select * from #TableA order by VisitID, TheDay
From #Patients
```

Creating Efficiencies

- Indexing and Primary Keys
  - Execution Plan
- Joining on Primary Keys
- Filters
- Where Exists
- Functions

Acmeware


Microsoft SQL Server Management Studio
File Edit View Project Debug Tools Window Communit
New Query
Execute
Object Explorer
Connect

```
FOXTROT\DEV (SQL Server 11.0.5058 - HQ\jgerardo)
  Databases
    System Databases
    Database Snapshots
    AcmewareLive
    Admission_Demo
    AdventureWorksDW2012
    livedb_CS_566
      Database Diagrams
      Tables (filtered)
        System Tables
        dbo.AbsAdmRcrDates
        dbo.AbsAdmServices
        dbo.AbsAdmSpecialCareServices
        dbo.AbsCanVisitAdmDxs
        dbo.AdmAllergies
        dbo.AdmBedDetails
          Columns
          Keys
          Constraints
          Triggers
          Indexes
            mtpk_cs551666 (Clustered)
          Statistics
        dbo.AdmBedRequestAttributes
        dbo.AdmBedRequestQueries
        dbo.AdmBedRequestQueriesMult
        dbo.AdmBedRequestReportAudits
        dbo.AdmBedRequestReports
```

Table Structure

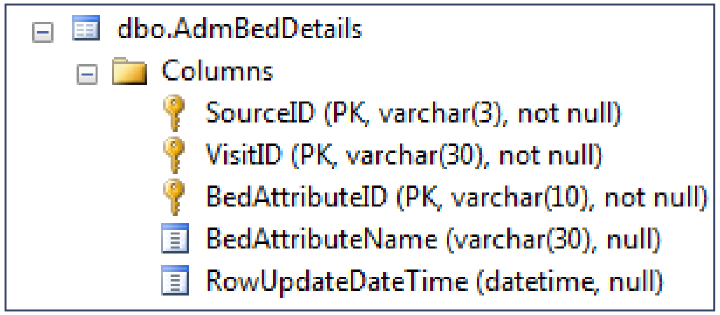
- Each MEDITECH table is implemented with one Index – the tables clustered index.
- Additional indexes can be built to increase query efficiency. (and should be)

Table Indexing

- Clustered
  - Every MEDITECH table has a clustered index, which is the physical order of the table by **primary key(s).** Never modify or delete
  - There is only 1 per table
- Non-Clustered
  - A non-clustered index creates a separate 'internal' table that stores only the selected key values of the table in order. Each 'record' in this index contains the key value from one record in the table along with a pointer to either the data record itself or to a value in the clustered index.

What are primary keys?

- Fields (columns) in a table that are special.
- The primary key values make a record unique to the table.
- Every MEDITECH table will have at least two primary keys per table. SourceID is always the first key.



```
dbo.AdmBedDetails
  Columns
    SourceID (PK, varchar(3), not null)
    VisitID (PK, varchar(30), not null)
    BedAttributeID (PK, varchar(10), not null)
    BedAttributeName (varchar(30), null)
    RowUpdateDateTime (datetime, null)
```

Common Table indexes

livedb

- BarChargeTransactions
  - Ix_ServiceDateTime
  - Ix_TransactionProcedureID
  - Ix_ProcedureChargeDept
- BarVisits
  - Ix_VisitID
  - Ix_AdmitDateTime
  - Ix_ServiceDateTime
- AdmVisits
  - Ix_ServiceDateTime
  - Ix_Status
- BarCollectionTransactions
  - Ix_ReceiptDateTime
  - Ix_InsuranceID
- AdmittingData
  - Ix_AdmitDateTime
- AbstractData
  - Ix_VisitID
- DMisUserStatisticsDetail
  - Ix_AccountNumber (Field4)
  - Ix_UnitNumber (Field3)

livefdb

- RegAcctQuery_Results
  - ix_DateTime
  - ix_InstanceID
  - ix_Query_MisQryID
- RegAcct_Main
  - ix_ArrivalDateTime
  - ix_ServiceDateTime
  - ix_AdmitDateTime
- OmOrd_Main
  - ix_SourceID_VisitID
  - ix_OrderDateTime

ware

Creating an Index

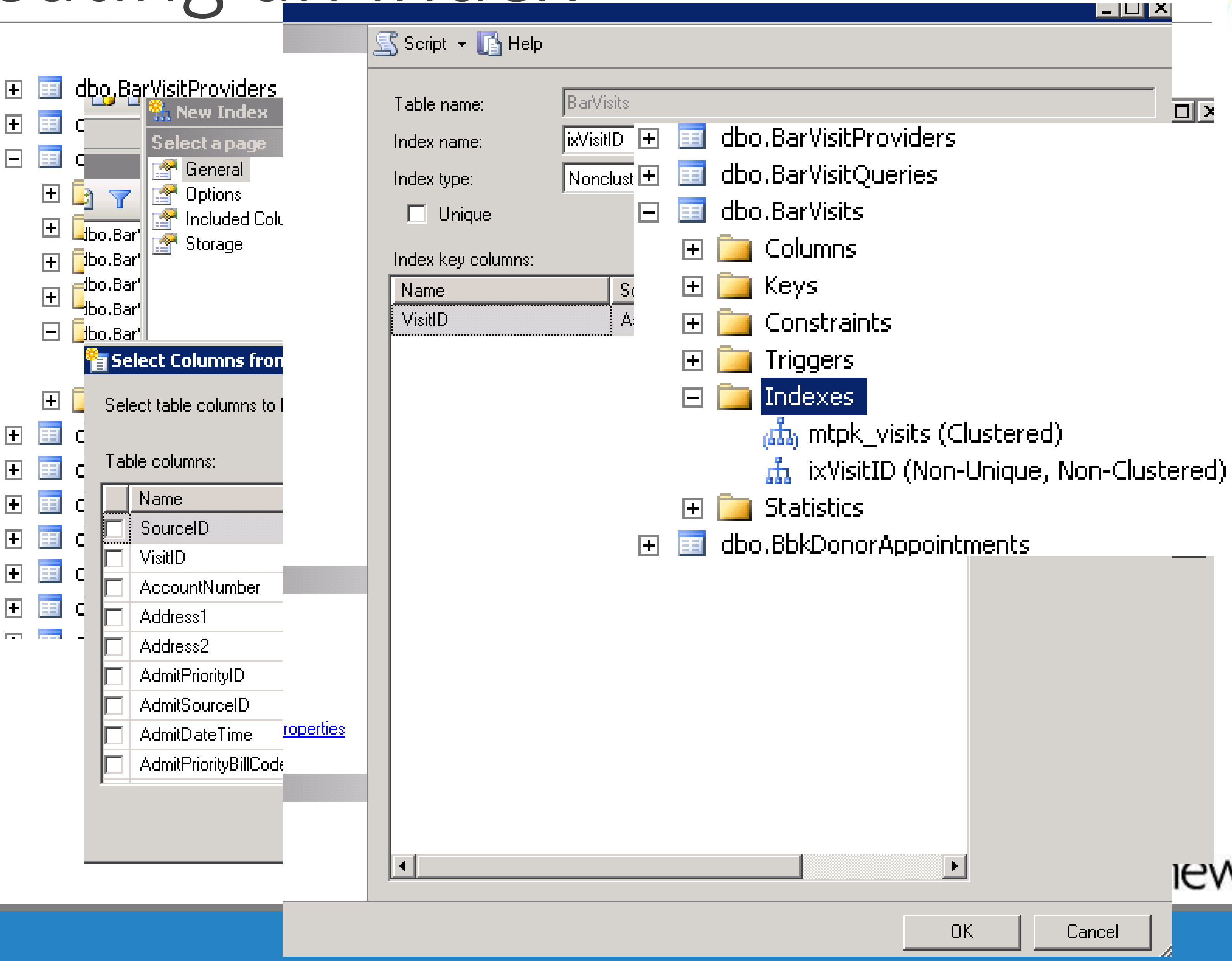


Table name: BarVisits
Index name: ixVisitID
Index type: Nonclust
Unique

Index key columns:

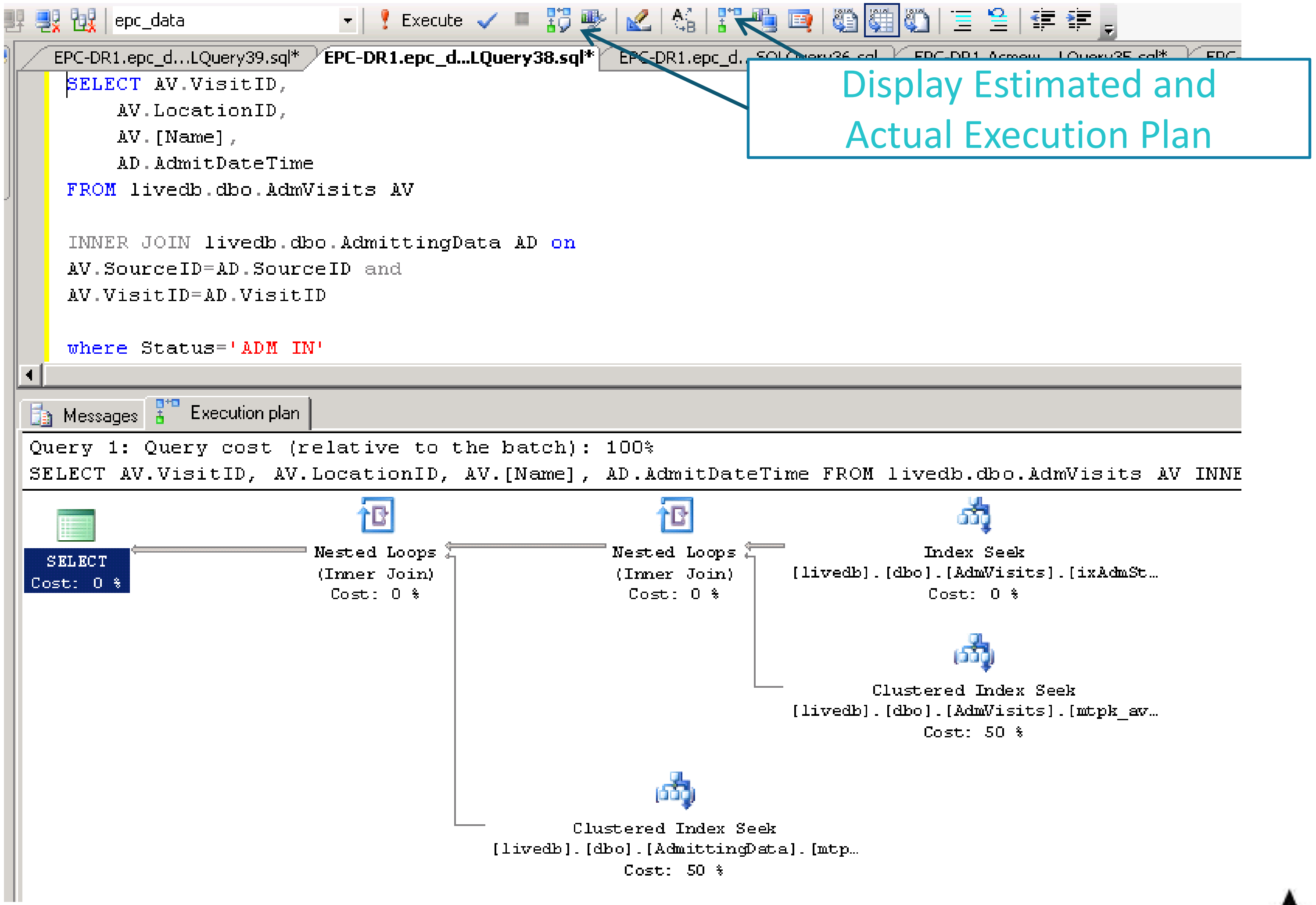
Name	So
VisitID	A

Table columns:

	Name
☐	SourceID
☐	VisitID
☐	AccountNumber
☐	Address1
☐	Address2
☐	AdmitPriorityID
☐	AdmitSourceID
☐	AdmitDateTime
☐	AdmitPriorityBillCode

- dbo.BarVisitProviders
- dbo.BarVisitQueries
- dbo.BarVisits
  - Columns
  - Keys
  - Constraints
  - Triggers
  - Indexes
    - mtpk_visits (Clustered)
    - ixVisitID (Non-Unique, Non-Clustered)
  - Statistics
- dbo.BbkDonorAppointments

Example of Execution Plan


epc_data
Execute
EPC-DR1.epc_d...LQuery39.sql*
EPC-DR1.epc_d...LQuery38.sql*
EPC-DR1.epc_d...SQLQuery36.sql
EPC-DR1.Acmew...LQuery35.sql*
Display Estimated and Actual Execution Plan
Messages
Execution plan

```
SELECT AV.VisitID,
       AV.LocationID,
       AV.[Name],
       AD.AdmitDateTime
FROM livedb.dbo.AdmVisits AV

INNER JOIN livedb.dbo.AdmittingData AD on
AV.SourceID=AD.SourceID and
AV.VisitID=AD.VisitID

where Status='ADM IN'
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT AV.VisitID, AV.LocationID, AV.[Name], AD.AdmitDateTime FROM livedb.dbo.AdmVisits AV INNE
```

SELECT
Cost: 0 %
Nested Loops
(Inner Join)
Cost: 0 %
Nested Loops
(Inner Join)
Cost: 0 %
Index Seek
[livedb].[dbo].[AdmVisits].[ixAdmSt…
Cost: 0 %
Clustered Index Seek
[livedb].[dbo].[AdmVisits].[mtpk_av…
Cost: 50 %
Clustered Index Seek
[livedb].[dbo].[AdmittingData].[mtp…
Cost: 50 %

Primary Keys

- Joining on the primary keys will make your report run more efficiently.
- Omitting the primary key could slow down your query and can skew your intended output.
- Each application has a unique identifier (primary key) that will allow you to join to other applications.
- All primary keys will end in either ID or DateTime
- In the M-AT 6.1 release, VisitID is the most commonly used primary to join from one application to another using the _Main tables
- Typically, all primary keys should be addressed in your Query

Acmeware

Primary Key Example

```
SELECT AV.VisitID,
              AV.LocationID,
              AV.[Name],
              AD.AdmitDateTime
FROM livedb.dbo.AdmVisits AV WITH (NOLOCK)

INNER JOIN livedb.dbo.AdmittingData AD WITH (NOLOCK)
ON AV.SourceID = AD.SourceID
AND AV.VisitID = AD.VisitID

WHERE Status='ADM IN‘
```

Even if there is only one SourceID, you will want to use the Clustered Index for faster processing.

Primary Key Example 2

```
SELECT AV.VisitID,
AV.LocationID,
		AV.[Name],
		AD.AdmitDateTime,
		BV.PrimaryInsuranceID,
		BVFD.Balance

FROM livedb.dbo.AdmVisits AV

INNER JOIN livedb.dbo.AdmittingData AD
ON AV.SourceID = AD.SourceID
AND AV.VisitID = AD.VisitID

LEFT JOIN livedb.dbo.BarVisits BV
ON AV.SourceID = BV.SourceID
AND AV.VisitID = BV.VisitID

LEFT JOIN livedb.dbo.BarVisitFinancialData BVFD
ON BV.SourceID = BVFD.SourceID
AND BV.BillingID = BVFD.BillingID
```

VisitID is in a number of tables but you'll only want to use it to join to a parent type table - BarVisits, AbstractData, Lab Specimens, SchAppointments

Use the application's primary key (unique identifier) within the application tables.

- Adm - VisitID
- Bar – BillingID
- Abs – AbstractID
- Oe – OrderID
- Sch – AppointmentID
- Reg - VisitID

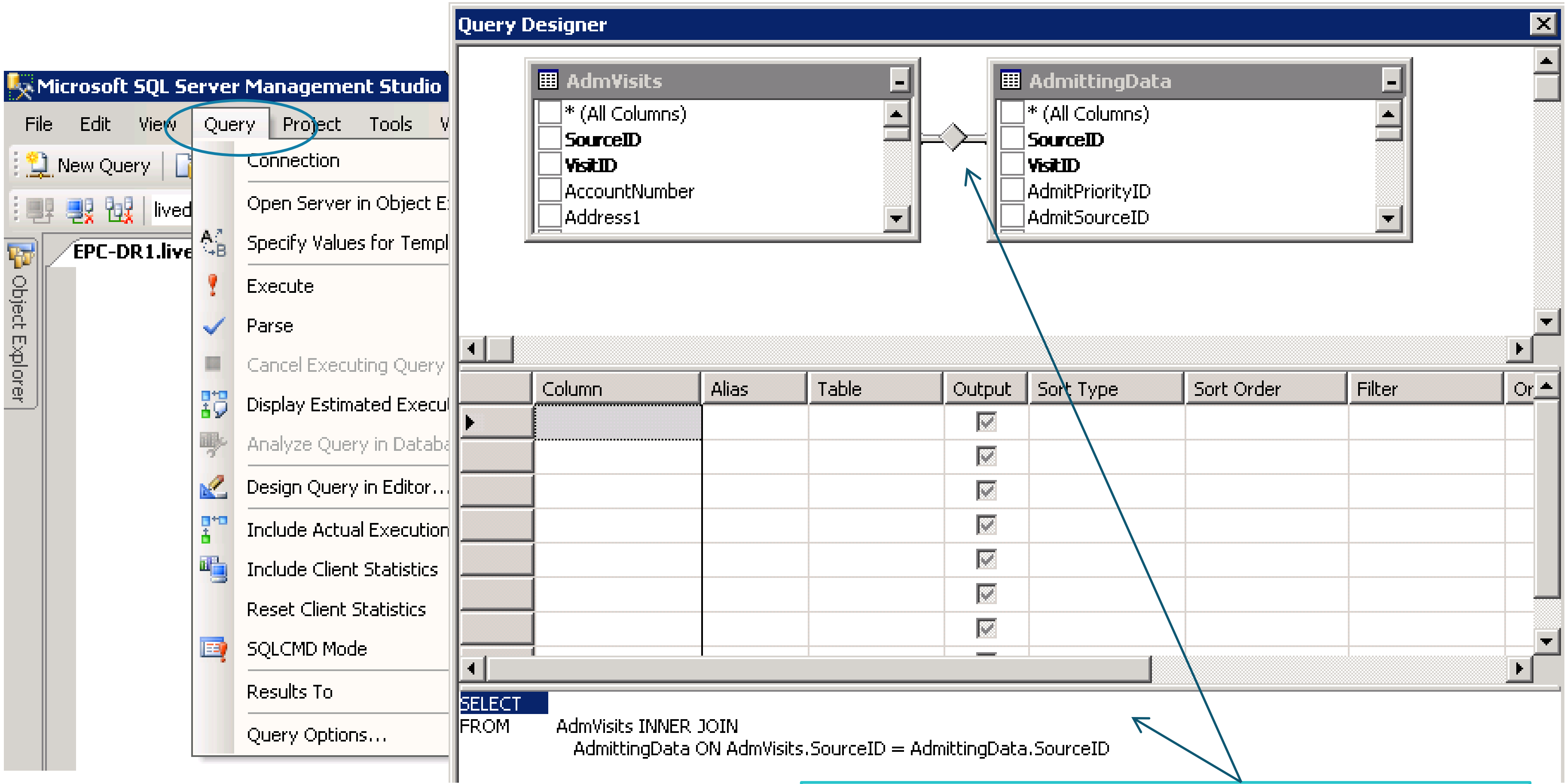
Application Parent tables (with patient data)

Platform	Application	Parent tables (patient data)	PrimaryKey To use within application tables	Foreign Key for joining from other applications	Notes
CS_Magic	ADM	AdmVisits	VisitID	VisitID or PatientID	
CS_Magic	BAR	BarVisits	BillingID	VisitID	
CS_Magic	ABS	AbstractData	AbstractID	VisitID	
CS_Magic	LAB	LabRequisitions	RequisitionID	VisitID	
CS_Magic	LAB	LabSpecimens	SpecimenID	VisitID	
CS	ITS	ItsOrders	OrderID	VisitID or OeOrderID	
CS_Magic	OE	OeOrders	OrderID	VisitID	
MAT	OM	OmOrd_Main	OmOrdID	VisitID or PatientID	
CS_Magic	PHA	PhaRx	PrescriptionID	VisitID	
Magic	RAD	RadExams	PatientID	PatientID	* This is one of the exceptions
CS_Magic	SCH	SchAppointments	AppointmentID	VisitID	
CS_Magic	SCH	SchPatOrCaseMain	CaseID	VisitID or PatientID	* Patient may not have VisitID

In the M-AT 6.+ releases, VisitID is the most commonly used primary to join from one application to another using the _Main tables

Acmeware

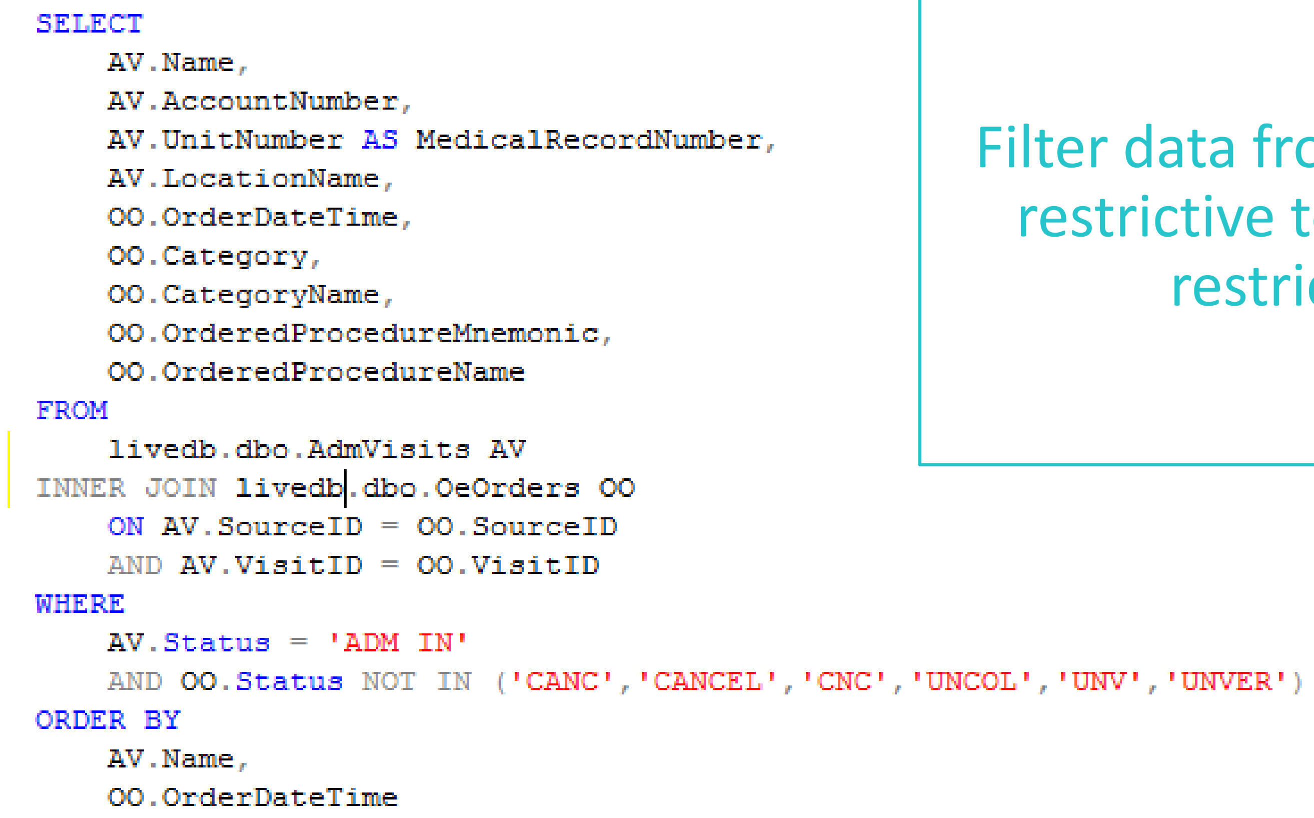
SQL Design Query Editor



```
SELECT
FROM        AdmVisits INNER JOIN
              AdmittingData ON AdmVisits.SourceID = AdmittingData.SourceID
```

You can use this tool but you still need to join on the primary keys. This tool will not automatically do that for you.

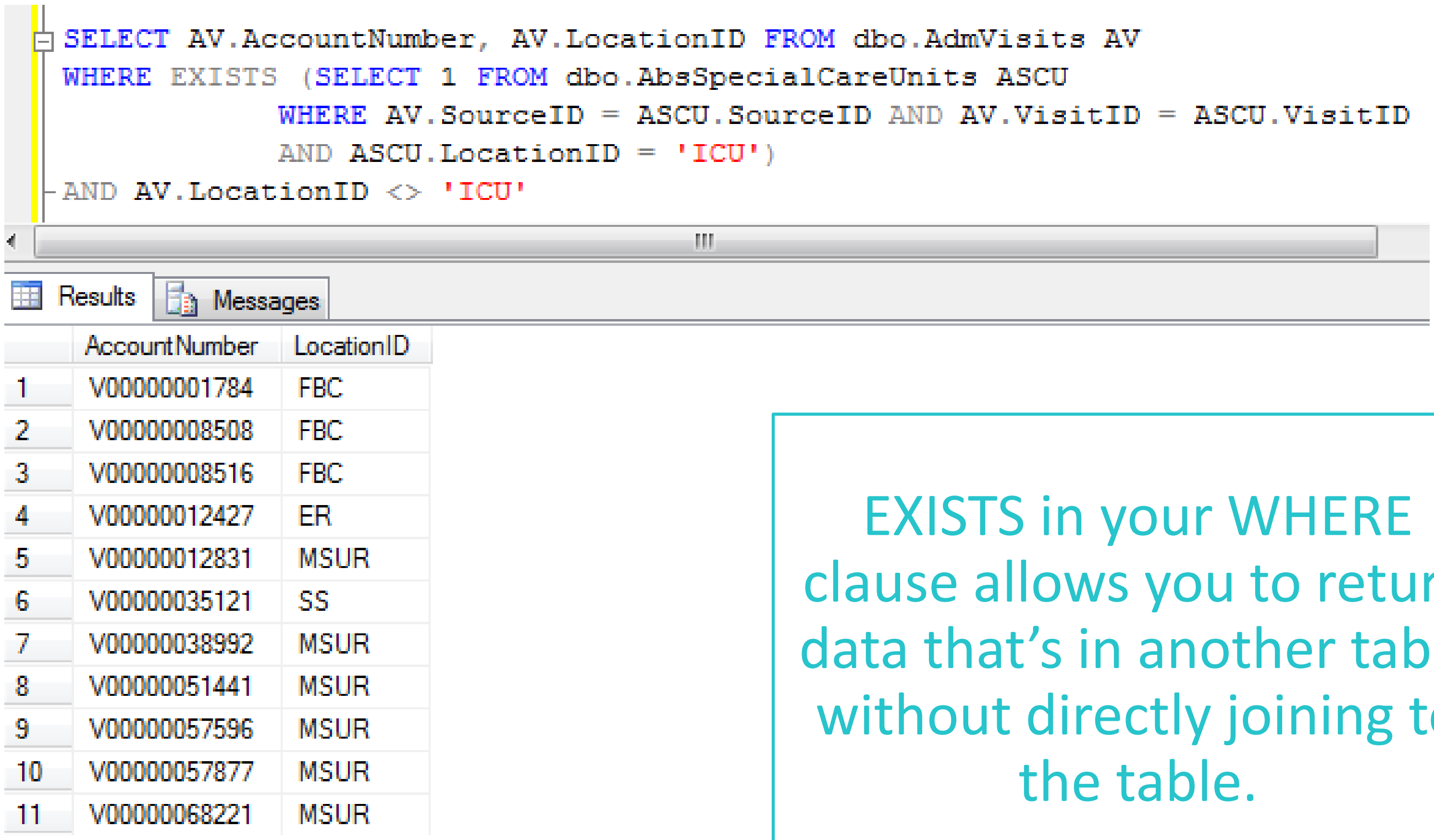
WHERE Clause (filtering your data)



```sql
SELECT
    AV.Name,
    AV.AccountNumber,
    AV.UnitNumber AS MedicalRecordNumber,
    AV.LocationName,
    OO.OrderDateTime,
    OO.Category,
    OO.CategoryName,
    OO.OrderedProcedureMnemonic,
    OO.OrderedProcedureName
FROM
    livedb.dbo.AdmVisits AV
INNER JOIN livedb.dbo.OeOrders OO
    ON AV.SourceID = OO.SourceID
    AND AV.VisitID = OO.VisitID
WHERE
    AV.Status = 'ADM IN'
    AND OO.Status NOT IN ('CANC','CANCEL','CNC','UNCOL','UNV','UNVER')
ORDER BY
    AV.Name,
    OO.OrderDateTime
```

Filter data from the most restrictive to the least restrictive

Using EXISTS



```
SELECT AV.AccountNumber, AV.LocationID FROM dbo.AdmVisits AV
WHERE EXISTS (SELECT 1 FROM dbo.AbsSpecialCareUnits ASCU
            WHERE AV.SourceID = ASCU.SourceID AND AV.VisitID = ASCU.VisitID
            AND ASCU.LocationID = 'ICU')
AND AV.LocationID <> 'ICU'
```

	AccountNumber	LocationID
1	V00000001784	FBC
2	V00000008508	FBC
3	V00000008516	FBC
4	V00000012427	ER
5	V00000012831	MSUR
6	V00000035121	SS
7	V00000038992	MSUR
8	V00000051441	MSUR
9	V00000057596	MSUR
10	V00000057877	MSUR
11	V00000068221	MSUR

EXISTS in your WHERE clause allows you to return data that's in another table without directly joining to the table.

User Defined Function

What is a User Defined Function?

Functions are subroutines used to encapsulate frequently performed logic. Any code that must perform the logic incorporated in a function can call the function rather than having to repeat all of the function logic.

- **Built-in functions** operate as defined in the Transact-SQL Reference and cannot be modified. The functions can be referenced only in Transact-SQL statements using the syntax defined in the Transact-SQL Reference.
  - Examples AVG, SUM, COUNT,DATEADD, DATEDIFF,NAME, ETC..

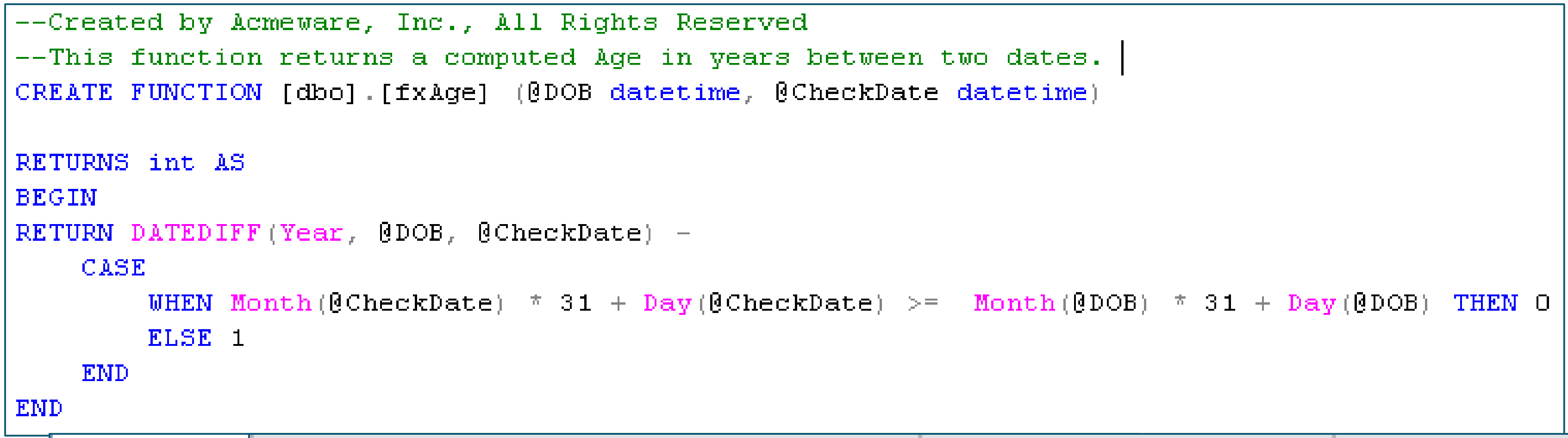
- **User-defined functions** allow you to define your own Transact-SQL functions using the CREATE FUNCTION statement. For more information about these built-in functions
  - This is what we'll looking at today.

FUNCTIONS

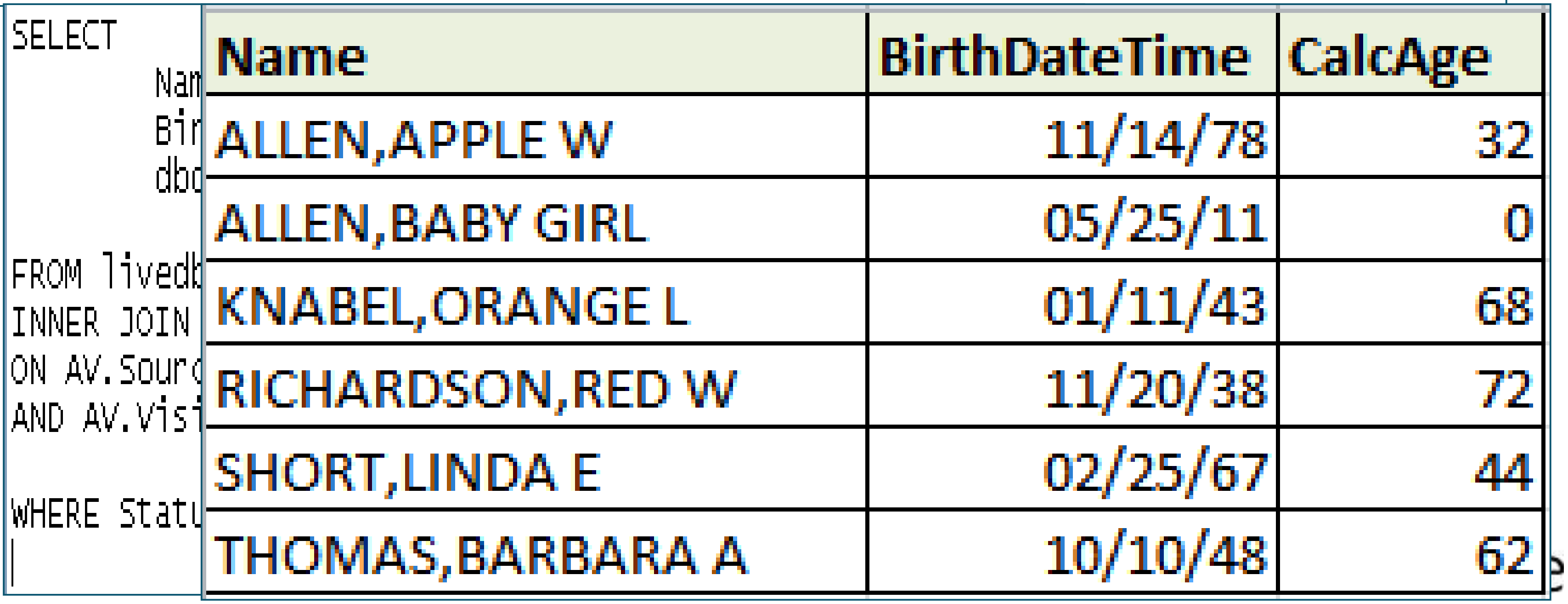
Useful Acmeware functions

- fxAge
- fxProperCase
- fxConvertGramsToLbs
- fxMeditechTimeIDToDateTime
- fxIsNumeric

Function - fx.Age



```
--Created by Acmeware, Inc., All Rights Reserved
--This function returns a computed Age in years between two dates.
CREATE FUNCTION [dbo].[fxAge] (@DOB datetime, @CheckDate datetime)

RETURNS int AS
BEGIN
RETURN DATEDIFF(Year, @DOB, @CheckDate) -
    CASE
        WHEN Month(@CheckDate) * 31 + Day(@CheckDate) >=  Month(@DOB) * 31 + Day(@DOB) THEN 0
        ELSE 1
    END
END
```



```
SELECT
         Nam
         Bir
         dbo


FROM livedb
INNER JOIN
ON AV.Sourc
AND AV.Visi

WHERE Statu
```

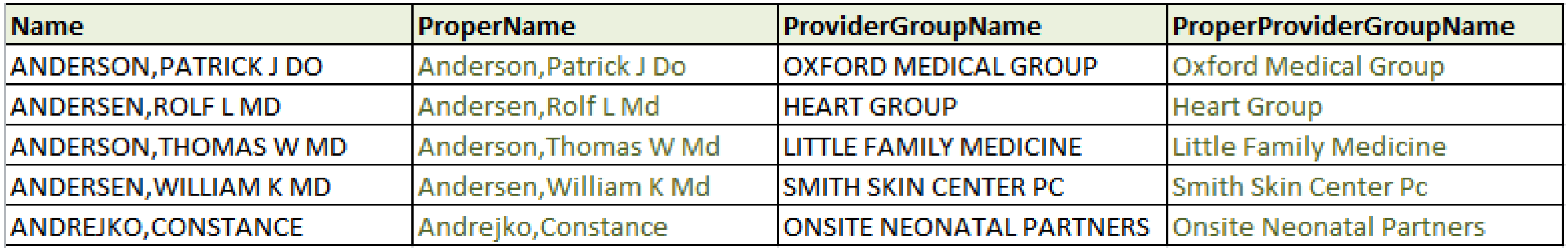
Name	BirthDateTime	CalcAge
ALLEN,APPLE W	11/14/78	32
ALLEN,BABY GIRL	05/25/11	0
KNABEL,ORANGE L	01/11/43	68
RICHARDSON,RED W	11/20/38	72
SHORT,LINDA E	02/25/67	44
THOMAS,BARBARA A	10/10/48	62

Function - fxProperCase

Selecting the data:

```
SELECT
            [Name],
            dbo.fxProperCase(Name) AS ProperName,
            ProviderGroupName,
            dbo.fxProperCase(ProviderGroupName)AS
                                    ProperGroupName
FROM livedb.dbo.DMisProvider
```

This takes any value and converts it to upper and lower case. Works great for creating consistencies in your reports.

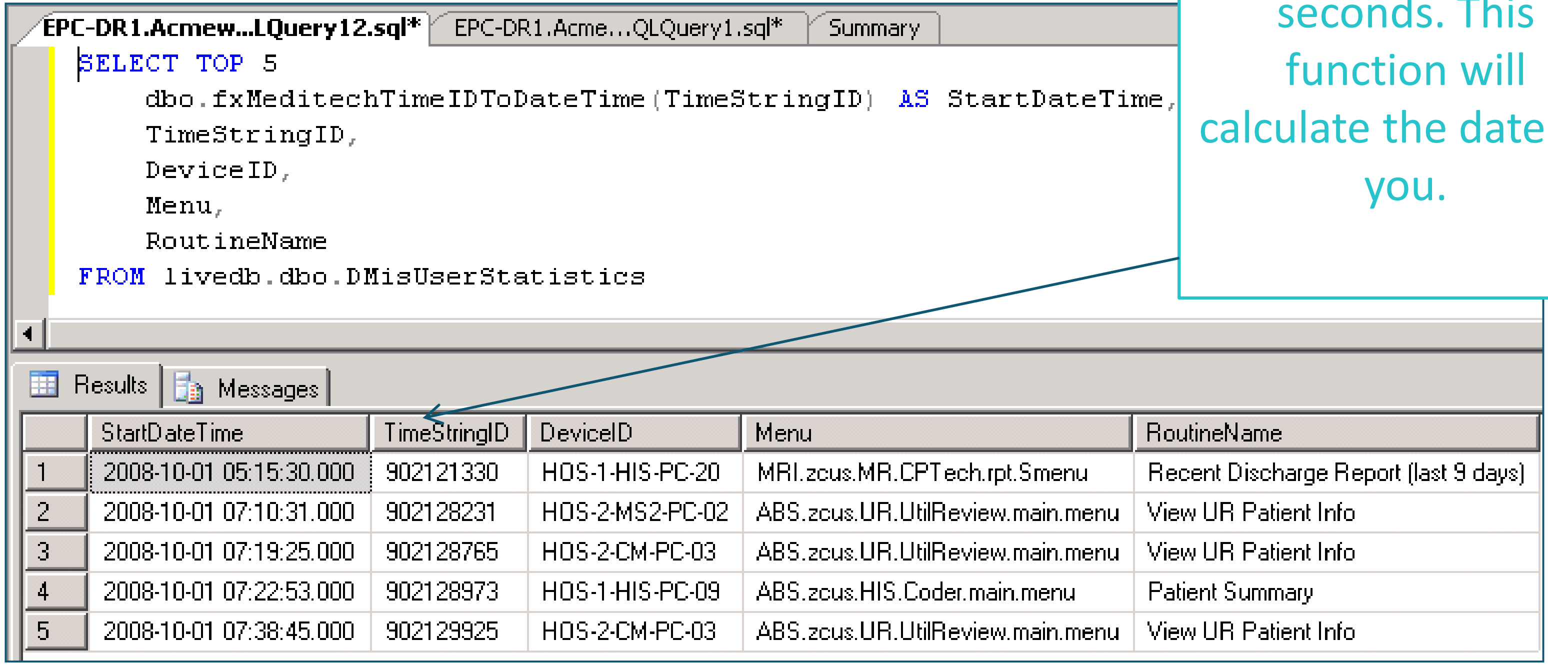


Name	ProperName	ProviderGroupName	ProperProviderGroupName
ANDERSON,PATRICK J DO	Anderson,Patrick J Do	OXFORD MEDICAL GROUP	Oxford Medical Group
ANDERSEN,ROLF L MD	Andersen,Rolf L Md	HEART GROUP	Heart Group
ANDERSON,THOMAS W MD	Anderson,Thomas W Md	LITTLE FAMILY MEDICINE	Little Family Medicine
ANDERSEN,WILLIAM K MD	Andersen,William K Md	SMITH SKIN CENTER PC	Smith Skin Center Pc
ANDREJKO,CONSTANCE	Andrejko,Constance	ONSITE NEONATAL PARTNERS	Onsite Neonatal Partners

Function - fxMeditechTimeIDToDateTime

```
--Created by Acmeware, Inc., All Rights Reserved
ALTER  FUNCTION [dbo].[fxMeditechTimeIDToDateTime] (@TimeID int)
RETURNS DATETIME AS
BEGIN
    RETURN  DATEADD(ss,CONVERT(Numeric,@TimeID),'3/1/1980')
END
```

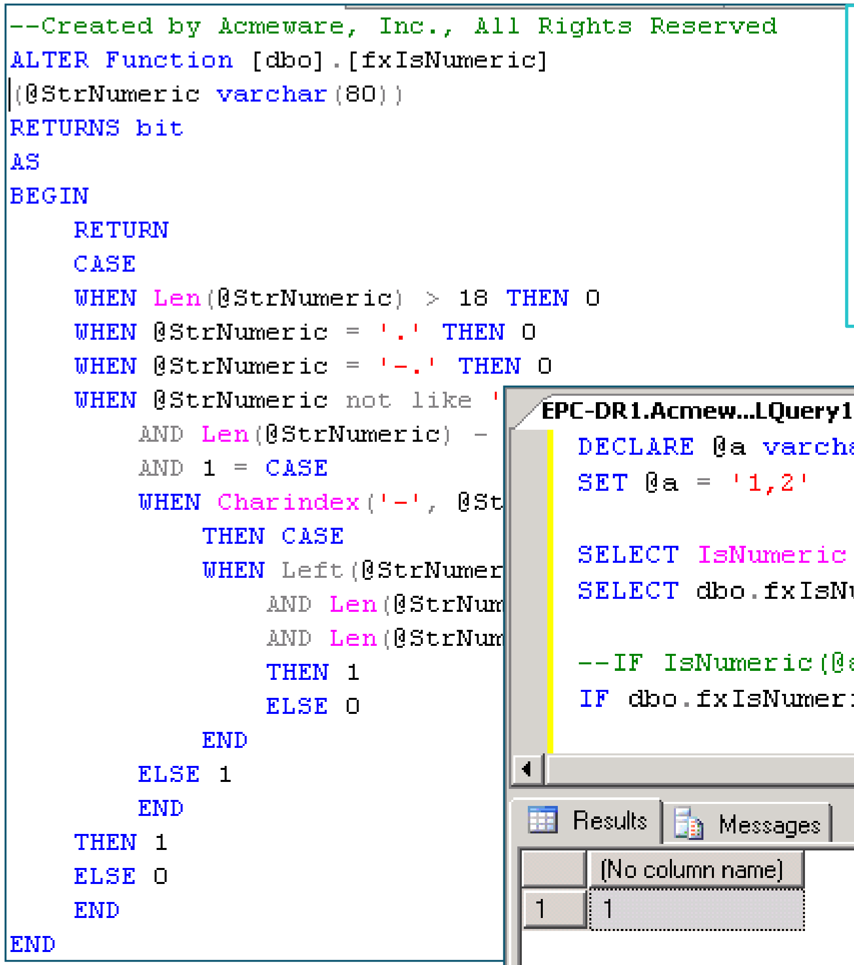
There are various fields throughout Meditech that are in seconds. This function will calculate the date for you.



```
SELECT TOP 5
    dbo.fxMeditechTimeIDToDateTime(TimeStringID) AS StartDateTime,
    TimeStringID,
    DeviceID,
    Menu,
    RoutineName
FROM livedb.dbo.DMisUserStatistics
```

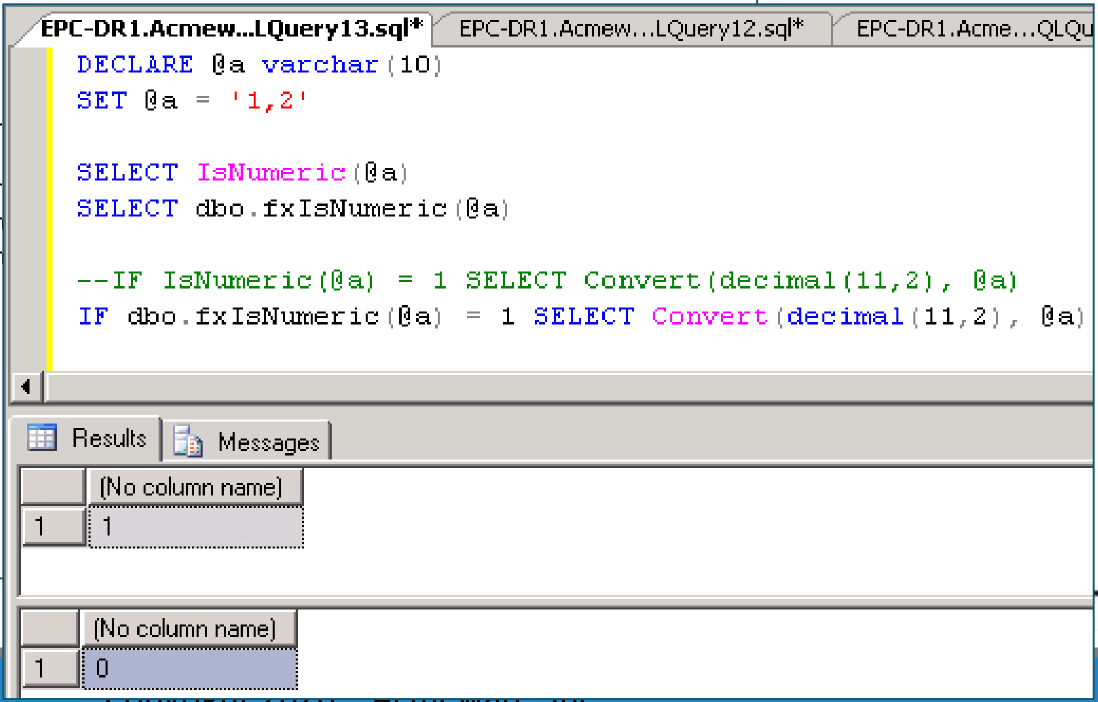
	StartDateTime	TimeStringID	DeviceID	Menu	RoutineName
1	2008-10-01 05:15:30.000	902121330	HOS-1-HIS-PC-20	MRI.zcus.MR.CPTech.rpt.Smenu	Recent Discharge Report (last 9 days)
2	2008-10-01 07:10:31.000	902128231	HOS-2-MS2-PC-02	ABS.zcus.UR.UtilReview.main.menu	View UR Patient Info
3	2008-10-01 07:19:25.000	902128765	HOS-2-CM-PC-03	ABS.zcus.UR.UtilReview.main.menu	View UR Patient Info
4	2008-10-01 07:22:53.000	902128973	HOS-1-HIS-PC-09	ABS.zcus.HIS.Coder.main.menu	Patient Summary
5	2008-10-01 07:38:45.000	902129925	HOS-2-CM-PC-03	ABS.zcus.UR.UtilReview.main.menu	View UR Patient Info

Function - fxIsNumeric



```
--Created by Acmeware, Inc., All Rights Reserved
ALTER Function [dbo].[fxIsNumeric]
(@StrNumeric varchar(80))
RETURNS bit
AS
BEGIN
    RETURN
    CASE
    WHEN Len(@StrNumeric) > 18 THEN 0
    WHEN @StrNumeric = '.' THEN 0
    WHEN @StrNumeric = '-.' THEN 0
    WHEN @StrNumeric not like '
        AND Len(@StrNumeric) -
        AND 1 = CASE
        WHEN Charindex('-', @St
            THEN CASE
            WHEN Left(@StrNumer
                AND Len(@StrNum
                AND Len(@StrNum
                THEN 1
                ELSE 0
            END
        ELSE 1
        END
    THEN 1
    ELSE 0
    END
END
```

There will be times where you need to ensure that a field strictly has numeric values. Using the System IsNumeric does not always work.



EPC-DR1.Acmew...LQuery13.sql* | EPC-DR1.Acmew...LQuery12.sql* | EPC-DR1.Acme...QLQu

```
DECLARE @a varchar(10)
SET @a = '1,2'

SELECT IsNumeric(@a)
SELECT dbo.fxIsNumeric(@a)

--IF IsNumeric(@a) = 1 SELECT Convert(decimal(11,2), @a)
IF dbo.fxIsNumeric(@a) = 1 SELECT Convert(decimal(11,2), @a)
```

Results | Messages

```
   (No column name)
1  1

   (No column name)
1  0
```

CTRL + Z

Remember you can always (almost) undo your last command

SQL Tips

- Temp Tables
- Row_Number
- Multiples to a single column
- Dates
- Using WITH (NOLOCK)

What is a temp table?

- There are two types of Temp tables:
  - Active within the same window - #TempTable
  - Active with your connection to the server - ##TempTable
- Temp tables are created on the fly to store data temporarily
- The temp tables are then joined to other SQL tables for further analysis or for calculating aggregates
- To avoid taking up excess space, you typically will not order data being put into a temp table (there are exceptions)
- Temp tables are deleted when the connection to the database is closed (query window is closed) or the table is dropped
  - CAUTION: When querying data, open SQL windows will retain the allocated space being used

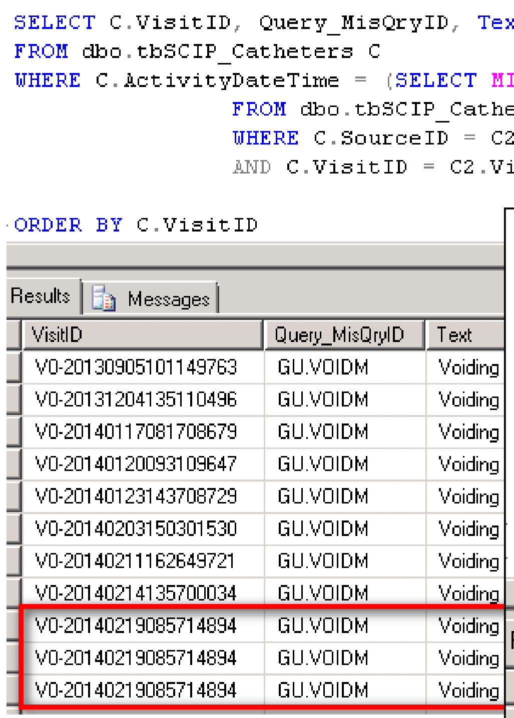
Code for Dropping Temp Tables

When using temp tables enter this before each temp table and it will save you a lot of time and hassle with continuously dropping the table.

```
IF OBJECT_ID('tempdb.dbo.#TableName')IS NOT NULL

DROP TABLE #TableName


SELECT

  Fields

INTO #TableName

FROM MyTables
```

You can also enter the code at the end of your stored procedure or query to make sure the temp table has been dropped.

Using ROW_NUMBER

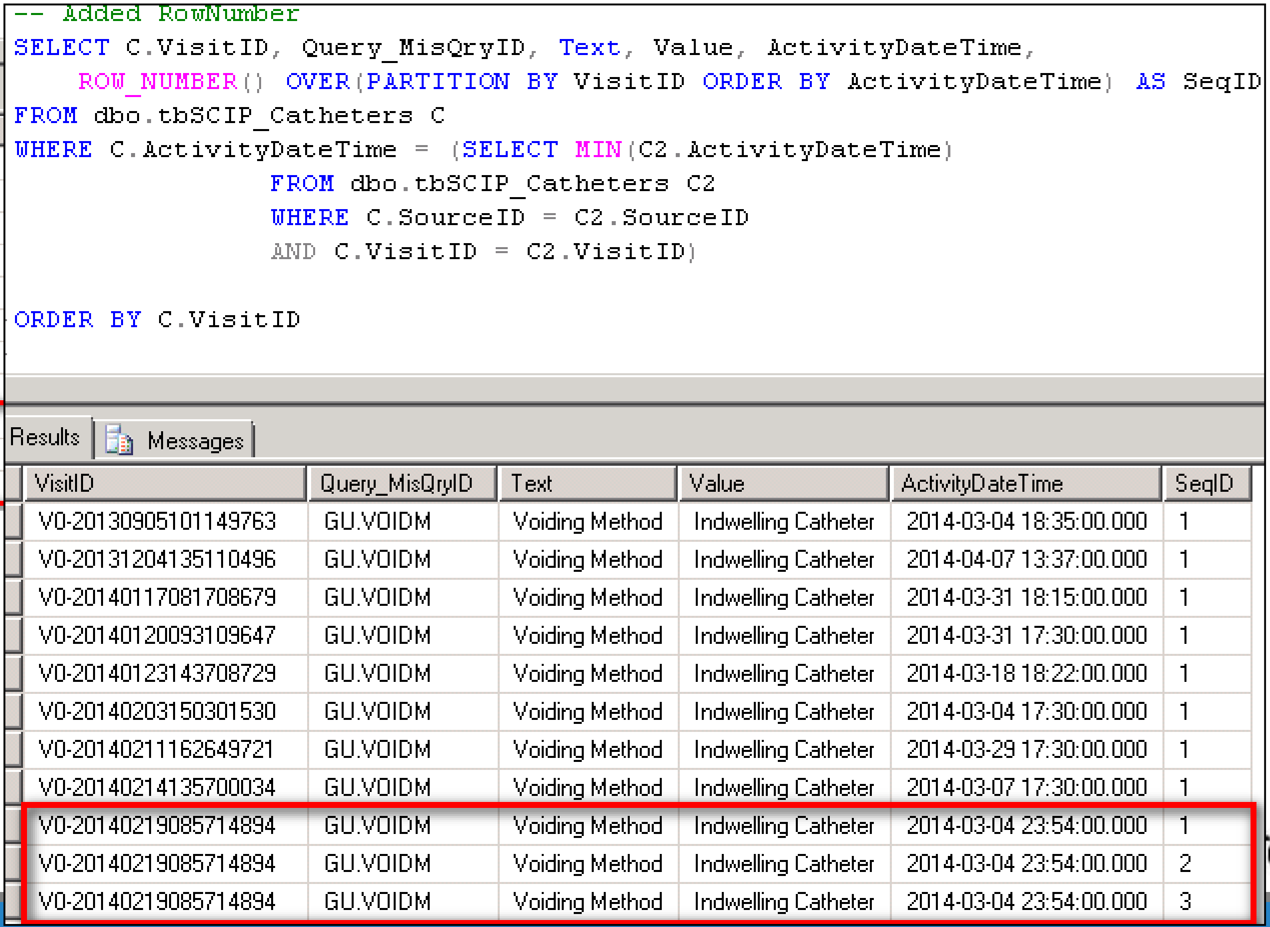


```
SELECT C.VisitID, Query_MisQryID, Text, Value, ActivityDateTime
FROM dbo.tbSCIP_Catheters C
WHERE C.ActivityDateTime = (SELECT MIN(C2.ActivityDateTime)
                FROM dbo.tbSCIP_Catheters C2
                WHERE C.SourceID = C2.SourceID
                AND C.VisitID = C2.VisitID)

ORDER BY C.VisitID
```

Results | Messages

VisitID	Query_MisQryID	Text
V0-20130905101149763	GU.VOIDM	Voiding
V0-20131204135110496	GU.VOIDM	Voiding
V0-20140117081708679	GU.VOIDM	Voiding
V0-20140120093109647	GU.VOIDM	Voiding
V0-20140123143708729	GU.VOIDM	Voiding
V0-20140203150301530	GU.VOIDM	Voiding
V0-20140211162649721	GU.VOIDM	Voiding
V0-20140214135700034	GU.VOIDM	Voiding
V0-20140219085714894	GU.VOIDM	Voiding
V0-20140219085714894	GU.VOIDM	Voiding
V0-20140219085714894	GU.VOIDM	Voiding

Creates a sequencing of rows based on field values.

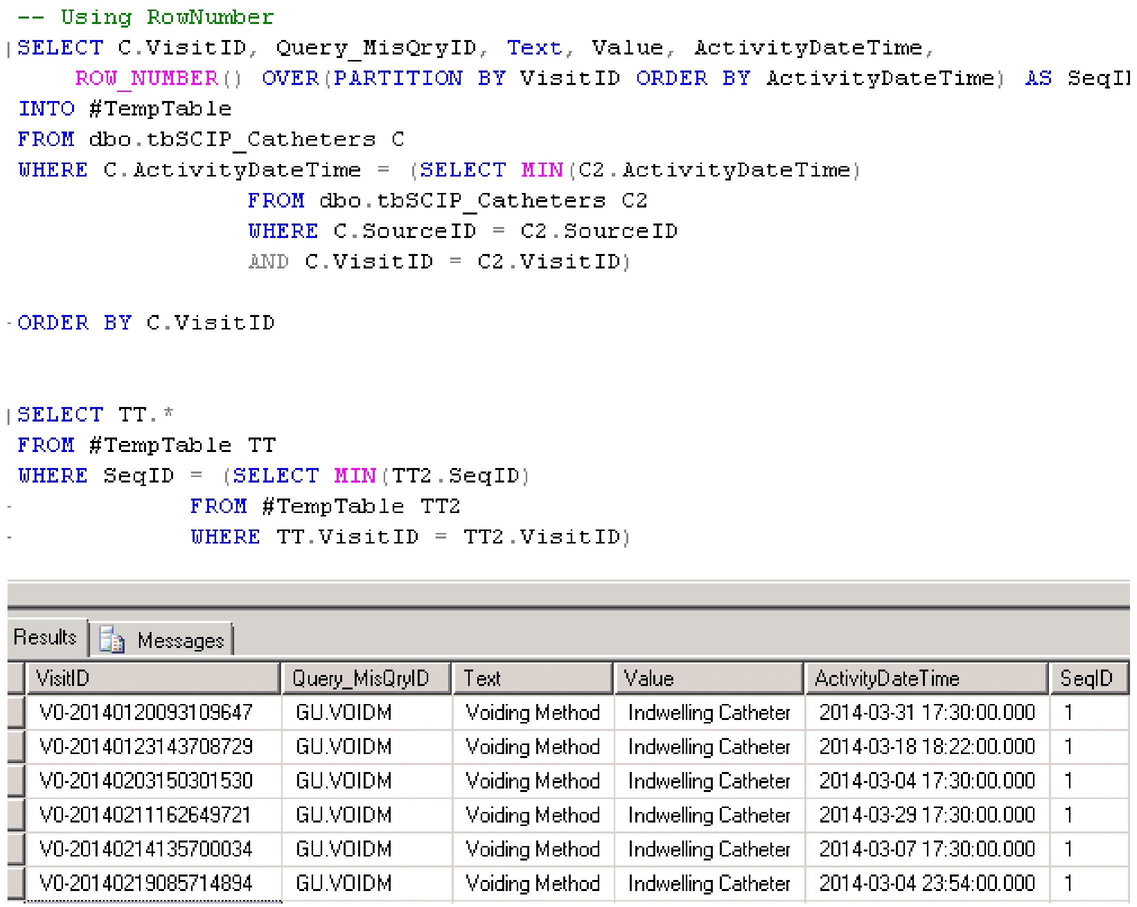


```
-- Added RowNumber
SELECT C.VisitID, Query_MisQryID, Text, Value, ActivityDateTime,
    ROW_NUMBER() OVER(PARTITION BY VisitID ORDER BY ActivityDateTime) AS SeqID
FROM dbo.tbSCIP_Catheters C
WHERE C.ActivityDateTime = (SELECT MIN(C2.ActivityDateTime)
                FROM dbo.tbSCIP_Catheters C2
                WHERE C.SourceID = C2.SourceID
                AND C.VisitID = C2.VisitID)

ORDER BY C.VisitID
```

Results | Messages

VisitID	Query_MisQryID	Text	Value	ActivityDateTime	SeqID
V0-20130905101149763	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-04 18:35:00.000	1
V0-20131204135110496	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-04-07 13:37:00.000	1
V0-20140117081708679	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-31 18:15:00.000	1
V0-20140120093109647	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-31 17:30:00.000	1
V0-20140123143708729	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-18 18:22:00.000	1
V0-20140203150301530	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-04 17:30:00.000	1
V0-20140211162649721	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-29 17:30:00.000	1
V0-20140214135700034	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-07 17:30:00.000	1
V0-20140219085714894	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-04 23:54:00.000	1
V0-20140219085714894	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-04 23:54:00.000	2
V0-20140219085714894	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-04 23:54:00.000	3

Output based on Row_Number field



```
-- Using RowNumber
SELECT C.VisitID, Query_MisQryID, Text, Value, ActivityDateTime,
    ROW_NUMBER() OVER(PARTITION BY VisitID ORDER BY ActivityDateTime) AS SeqI
INTO #TempTable
FROM dbo.tbSCIP_Catheters C
WHERE C.ActivityDateTime = (SELECT MIN(C2.ActivityDateTime)
                FROM dbo.tbSCIP_Catheters C2
                WHERE C.SourceID = C2.SourceID
                AND C.VisitID = C2.VisitID)

ORDER BY C.VisitID


SELECT TT.*
FROM #TempTable TT
WHERE SeqID = (SELECT MIN(TT2.SeqID)
            FROM #TempTable TT2
            WHERE TT.VisitID = TT2.VisitID)
```

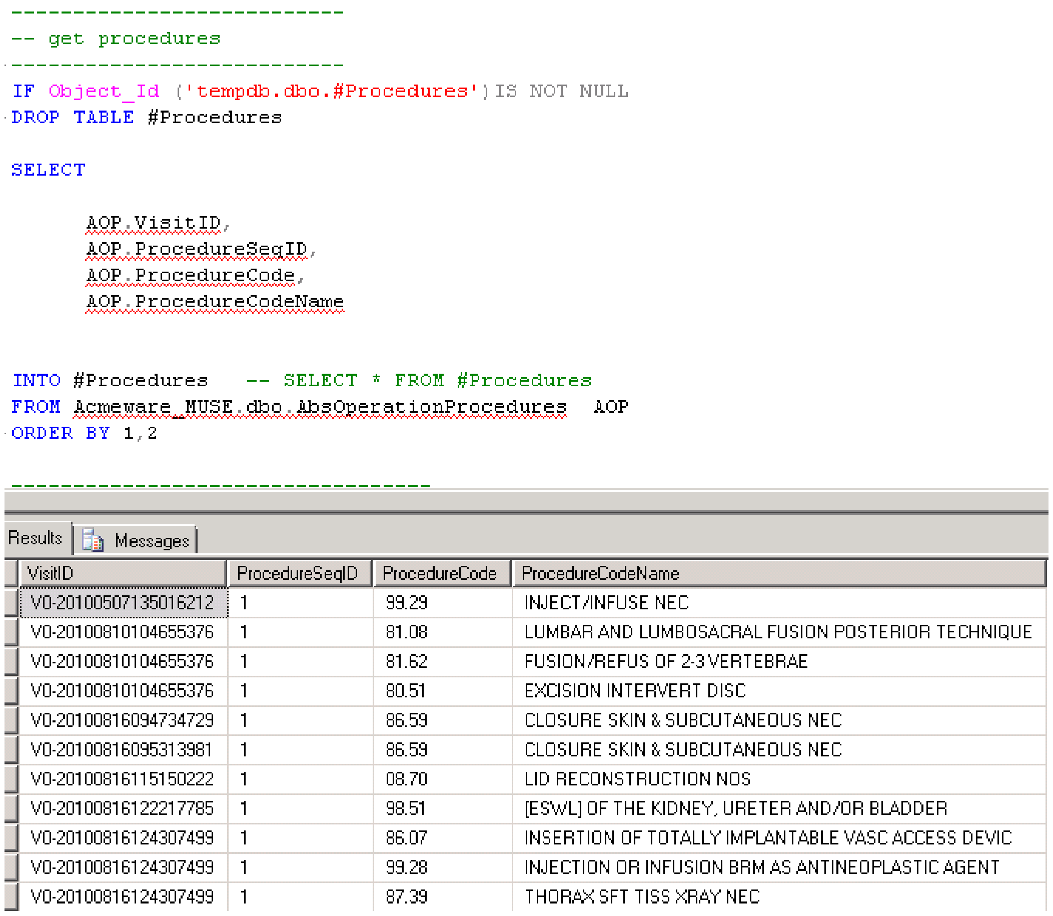
Results | Messages

VisitID	Query_MisQryID	Text	Value	ActivityDateTime	SeqID
V0-20140120093109647	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-31 17:30:00.000	1
V0-20140123143708729	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-18 18:22:00.000	1
V0-20140203150301530	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-04 17:30:00.000	1
V0-20140211162649721	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-29 17:30:00.000	1
V0-20140214135700034	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-07 17:30:00.000	1
V0-20140219085714894	GU.VOIDM	Voiding Method	Indwelling Catheter	2014-03-04 23:54:00.000	1

Sequencing rows are useful when your output needs to be a single row per patient, visit or other value.

Acmeware

Create a single column list
(from multiple value columns)



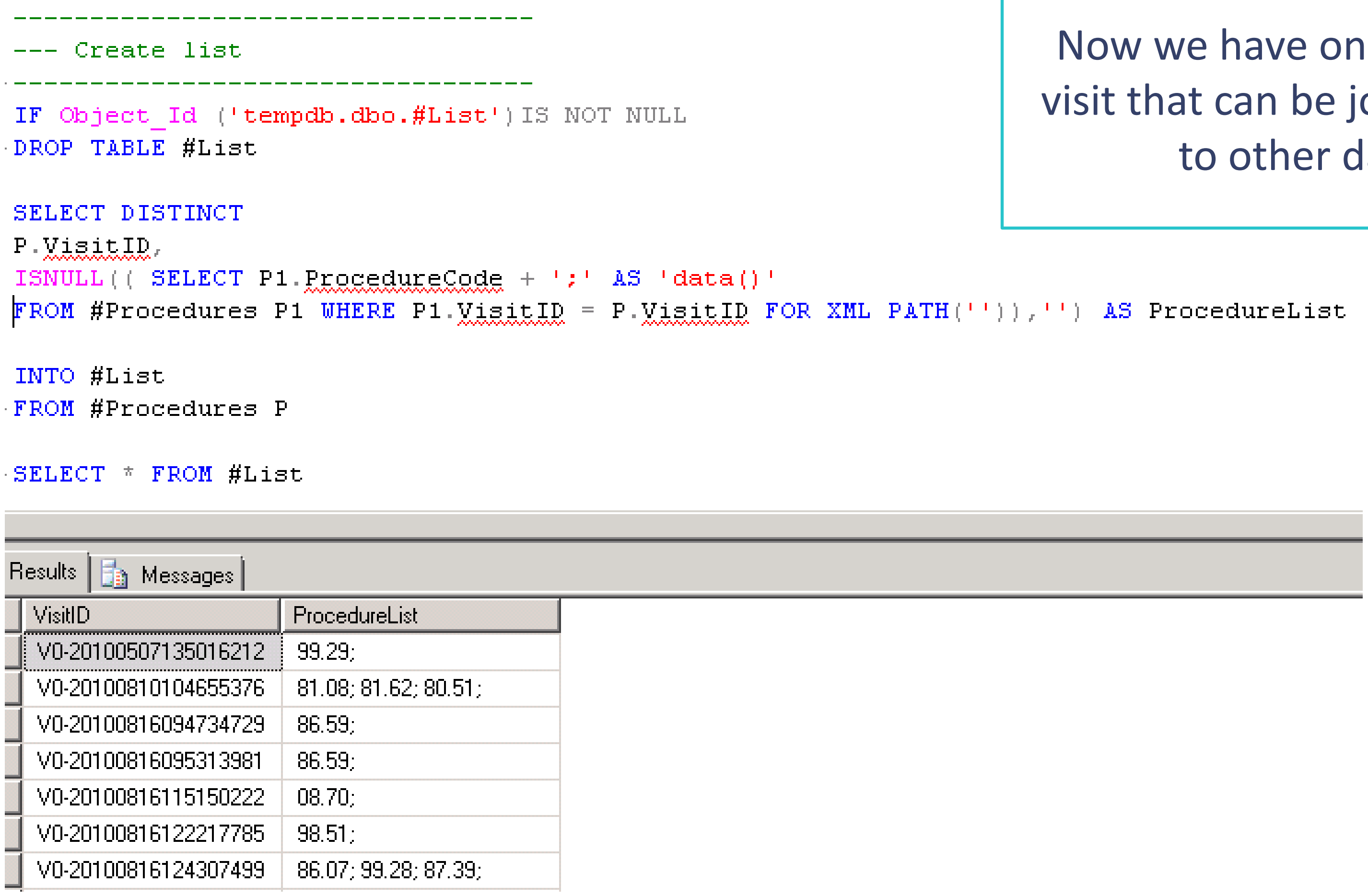
```
----------------------------
-- get procedures
----------------------------
IF Object_Id ('tempdb.dbo.#Procedures') IS NOT NULL
DROP TABLE #Procedures

SELECT

      AOP.VisitID,
      AOP.ProcedureSeqID,
      AOP.ProcedureCode,
      AOP.ProcedureCodeName


INTO #Procedures    -- SELECT * FROM #Procedures
FROM Acmeware_MUSE.dbo.AbsOperationProcedures  AOP
ORDER BY 1,2

-----------------------------------
```

Results | Messages

VisitID	ProcedureSeqID	ProcedureCode	ProcedureCodeName
V0-20100507135016212	1	99.29	INJECT/INFUSE NEC
V0-20100810104655376	1	81.08	LUMBAR AND LUMBOSACRAL FUSION POSTERIOR TECHNIQUE
V0-20100810104655376	1	81.62	FUSION/REFUS OF 2-3 VERTEBRAE
V0-20100810104655376	1	80.51	EXCISION INTERVERT DISC
V0-20100816094734729	1	86.59	CLOSURE SKIN & SUBCUTANEOUS NEC
V0-20100816095313981	1	86.59	CLOSURE SKIN & SUBCUTANEOUS NEC
V0-20100816115150222	1	08.70	LID RECONSTRUCTION NOS
V0-20100816122217785	1	98.51	[ESWL] OF THE KIDNEY, URETER AND/OR BLADDER
V0-20100816124307499	1	86.07	INSERTION OF TOTALLY IMPLANTABLE VASC ACCESS DEVIC
V0-20100816124307499	1	99.28	INJECTION OR INFUSION BRM AS ANTINEOPLASTIC AGENT
V0-20100816124307499	1	87.39	THORAX SFT TISS XRAY NEC

Some examples are cpt codes, diagnosis codes, procedure codes, and allergies

Using FOR XML to create a single list



```
-----------------------------------
--- Create list
-----------------------------------
IF Object_Id ('tempdb.dbo.#List')IS NOT NULL
DROP TABLE #List

SELECT DISTINCT
P.VisitID,
ISNULL(( SELECT P1.ProcedureCode + ';' AS 'data()'
FROM #Procedures P1 WHERE P1.VisitID = P.VisitID FOR XML PATH('')),'') AS ProcedureList

INTO #List
FROM #Procedures P

SELECT * FROM #List
```

Results | Messages

VisitID	ProcedureList
V0-20100507135016212	99.29;
V0-20100810104655376	81.08; 81.62; 80.51;
V0-20100816094734729	86.59;
V0-20100816095313981	86.59;
V0-20100816115150222	08.70;
V0-20100816122217785	98.51;
V0-20100816124307499	86.07; 99.28; 87.39;

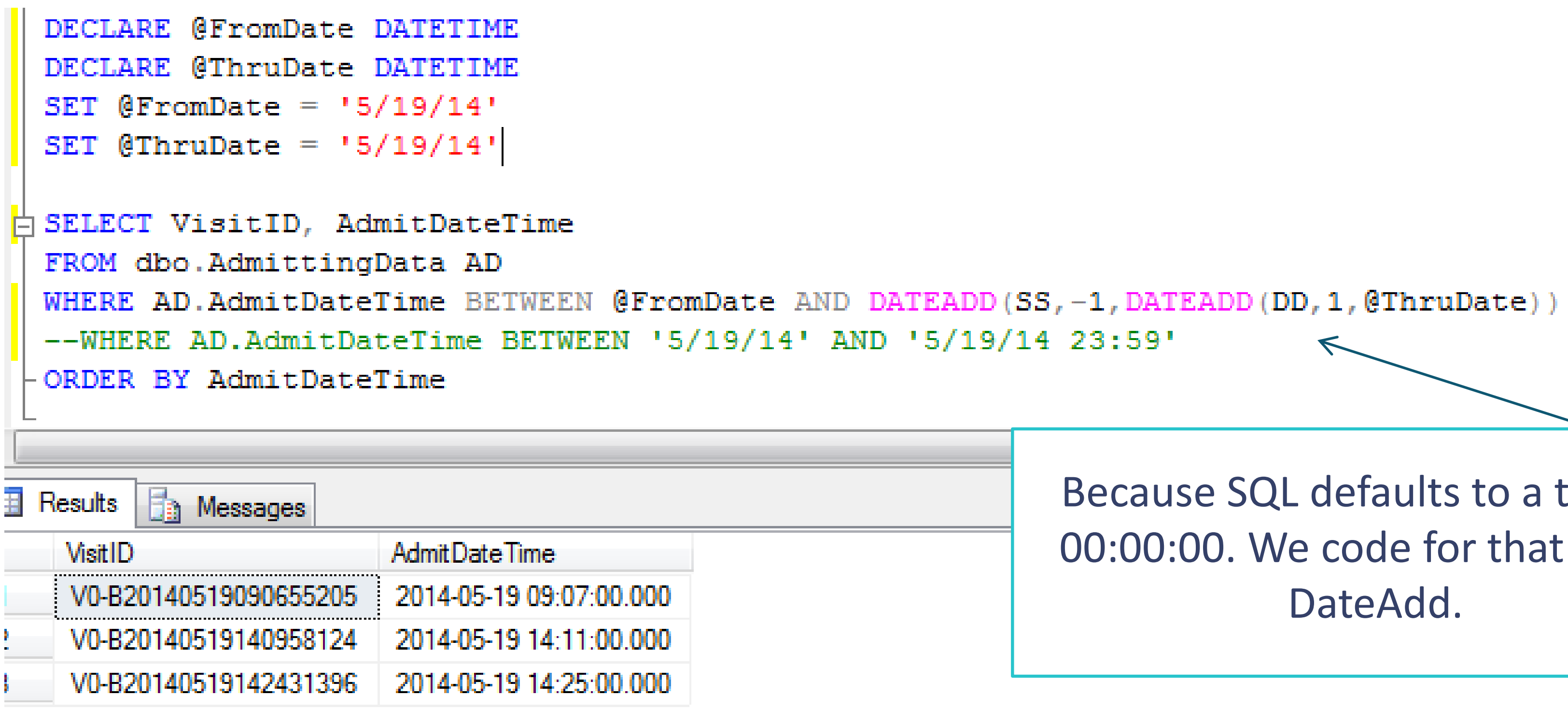
Now we have one row per visit that can be joined back to other data

TSQL Tips - Dates

- **SQL Date Default**
  - '5/26/17' defaults to 5/26/17 00:00:00
- **Getdate()**
  - Gets Current date and time
- **DateDiff**
  - Calculates the difference between two dates
- **DateAdd**
  - Adds a period of time to a date (or subtracts)
    - Years, Months, Days, Hours, Minutes or Seconds

These three functions will create any date you need to automate a stored procedure.

SQL Date Time Default



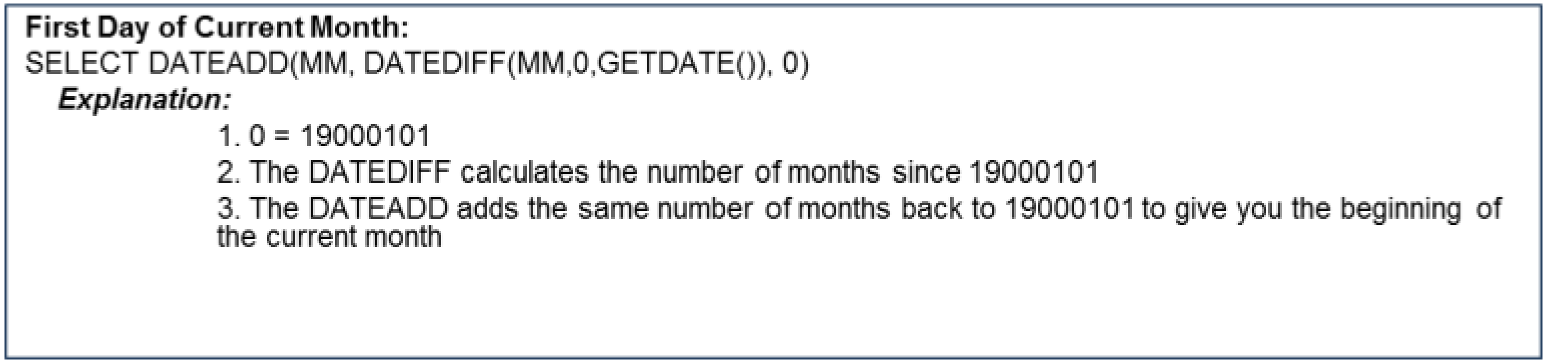
```
DECLARE @FromDate DATETIME
DECLARE @ThruDate DATETIME
SET @FromDate = '5/19/14'
SET @ThruDate = '5/19/14'

SELECT VisitID, AdmitDateTime
FROM dbo.AdmittingData AD
WHERE AD.AdmitDateTime BETWEEN @FromDate AND DATEADD(SS,-1,DATEADD(DD,1,@ThruDate))
--WHERE AD.AdmitDateTime BETWEEN '5/19/14' AND '5/19/14 23:59'
ORDER BY AdmitDateTime
```

VisitID	AdmitDateTime
V0-B20140519090655205	2014-05-19 09:07:00.000
V0-B20140519140958124	2014-05-19 14:11:00.000
V0-B20140519142431396	2014-05-19 14:25:00.000

Because SQL defaults to a time of 00:00:00. We code for that with a DateAdd.

Keep this in mind when creating data range parameters so that you include the full last day of the search

DateAdd Calculations



First Day of Current Month:
SELECT DATEADD(MM, DATEDIFF(MM,0,GETDATE()), 0)
Explanation:
1. 0 = 19000101
2. The DATEDIFF calculates the number of months since 19000101
3. The DATEADD adds the same number of months back to 19000101 to give you the beginning of the current month

Last Day of Current Month:
SELECT DATEADD(SS,-1,DATEADD(MM,DATEDIFF(MM,0,GETDATE())+1,0))
Explanation:
1. DATEDIFF(MM,0,GETDATE())+1 - calculates the number of months from the current date since 19000101 and adds 1
2. DATEADD(MM,DATEDIFF(MM,0,GETDATE())+1,0) - adds the above number of months to 19000101 (this will give you the first day of next month)
3. The last DATEADD substracts 1 second to give you the last day of the current month (ie. 9/30/09 23:59:59)

First Day of Last Month:
SELECT DATEADD(MM, DATEDIFF(MM,0,DATEADD(MM,-1,GETDATE())),0)
Explanation:
1. DATEADD(MM,-1,GETDATE()) - Subtracts 1 month from current date
2. DATEDIFF(MM,0,DATEADD(MM,-1,GETDATE())) - calculates the number of months since 19000101
3. The DATEADD adds the calculated number of months back to 19000101 to give you the beginning of the previous month

DateAdd Calculations

Last Day of Last Month:

```
SELECT DATEADD(SS,-1,DATEADD(MM,DATEDIFF(MM,0,GETDATE()),0))
```

Explanation:

DATEADD(MM,DATEDIFF(MM,0,GETDATE()),0) - same code as getting the first day of the current month

DATEADD substracts 1 second to give you the last day of previous month

First Day of Current Year:

```
SELECT DATEADD(YY,DATEDIFF(YY,0,GETDATE()),0)
```

Explanation:

1. 0 = 19000101
2. The DATEDIFF calculates the number of years since 19000101
3. The DATEADD adds the same number of years back to 19000101 to give you the beginning of the current year
4. This is the same as the month calculations but instead of mm for month you use the yy for year

Last Day of Last Year:

```
SELECT DATEADD(SS,-1,DATEADD(YY,DATEDIFF(YY,0,GETDATE()),0))
```

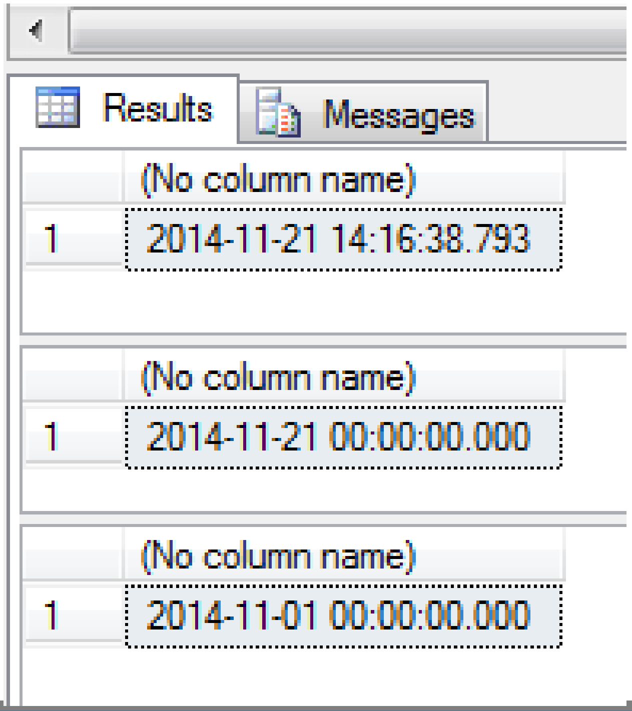
Explanation:

1. 0 = 19000101
2. The DATEDIFF calculates the number of years since 19000101
3. The DATEADD adds the same number of years back to 19000101 to give you the beginning of the current year
4. The next DATEADD substracts 1 second to reflect the day before just before midnight.

Acmeware

Examples using DateAdd

```
SELECT DATEADD(MM,-6,GETDATE())
          -- Subtracting 6 months from now
SELECT CONVERT(DATETIME,CONVERT(CHAR,DATEADD(MM,-6,GETDATE()),101))
          -- Subtracting 6 months from right now then removing time factor
SELECT DATEADD(MM,-6,DATEADD(MM,DATEDIFF(MM,0,GETDATE()),0))
          -- Getting the beginning of the month 6 months ago
```

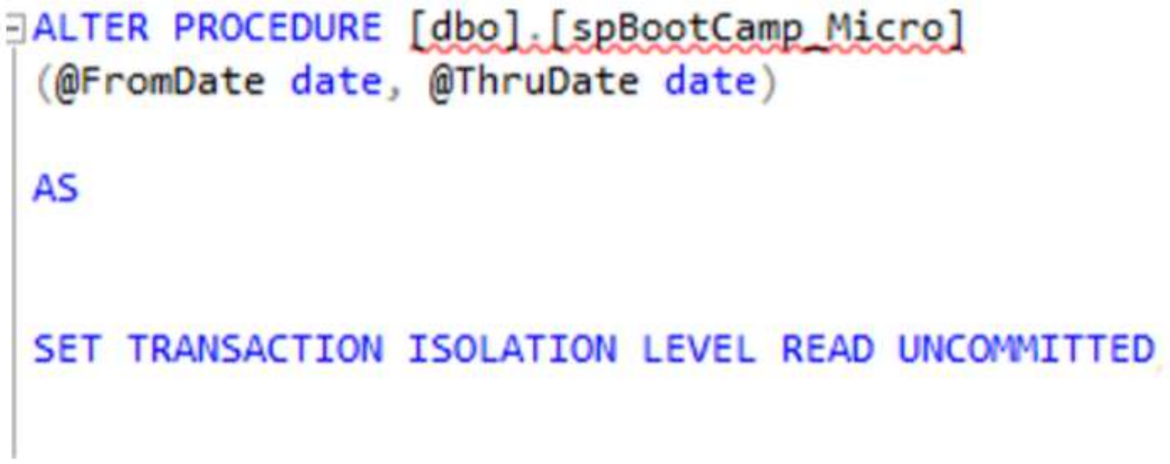


```
   (No column name)
1  2014-11-21 14:16:38.793

   (No column name)
1  2014-11-21 00:00:00.000

   (No column name)
1  2014-11-01 00:00:00.000
```

Understanding how the data functions work will help you write the appropriate code for your particular needs.

Using WITH (NOLOCK)

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED



```
ALTER PROCEDURE [dbo].[spBootCamp_Micro]
(@FromDate date, @ThruDate date)

AS


SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
```

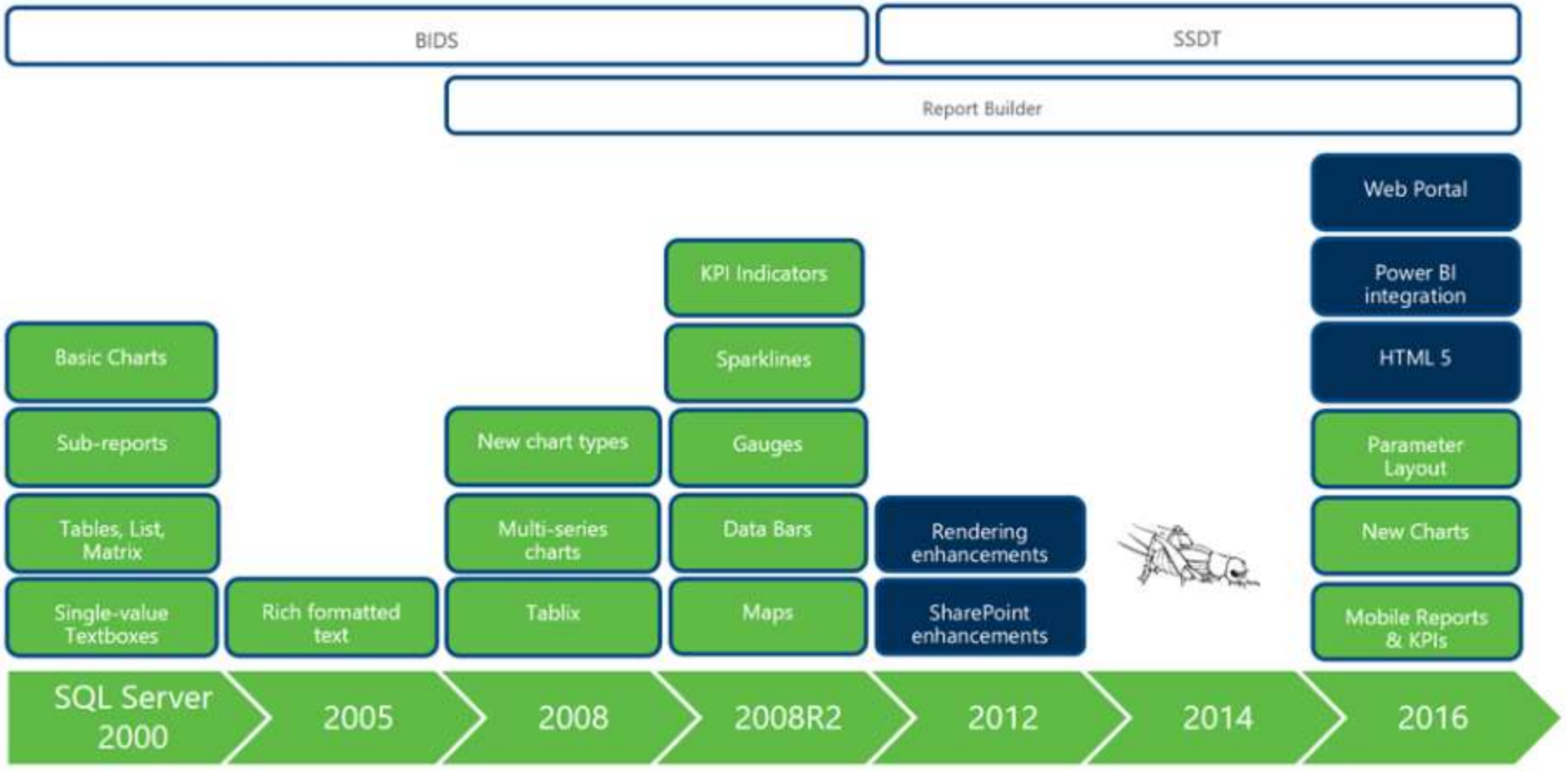
When data in a database is read or modified, the database engine uses special types of controls, called locks, to maintain integrity in the database. Locks basically work by making sure database records involved in a transaction cannot be modified by other transactions until the first transaction has committed, ensuring database consistency.

The benefit of using WITH (NOLOCK) is that it allows you to keep the database engine from issuing locks against the tables in your queries; this increases concurrency and performance because the database engine does not have to maintain the shared locks involved.

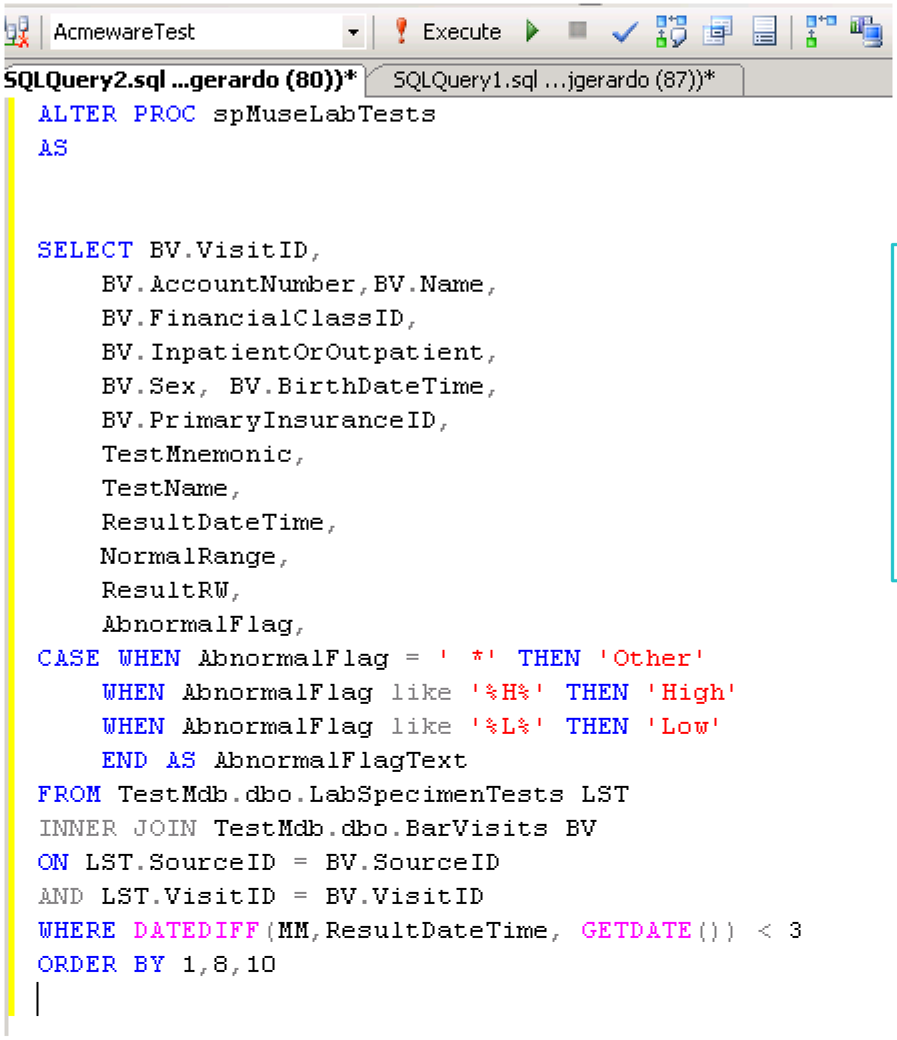
SSRS – Reporting Services

- Stored procedures and Reports are typically developed by someone in IS.
- The report is highly customizable with various options for display.
  - Tables, Matrix tables, charts and gauges are all reporting options.
- The reports are developed to run with or without input parameters.
- Reports are deployed and access given to groups and users
- End Users access and run the report but can not modify.
- Modifications are done in IS.
- SSRS Reports are the best option for more complicated SQL queries.

Reporting Services Feature Waves



Example Stored Procedure

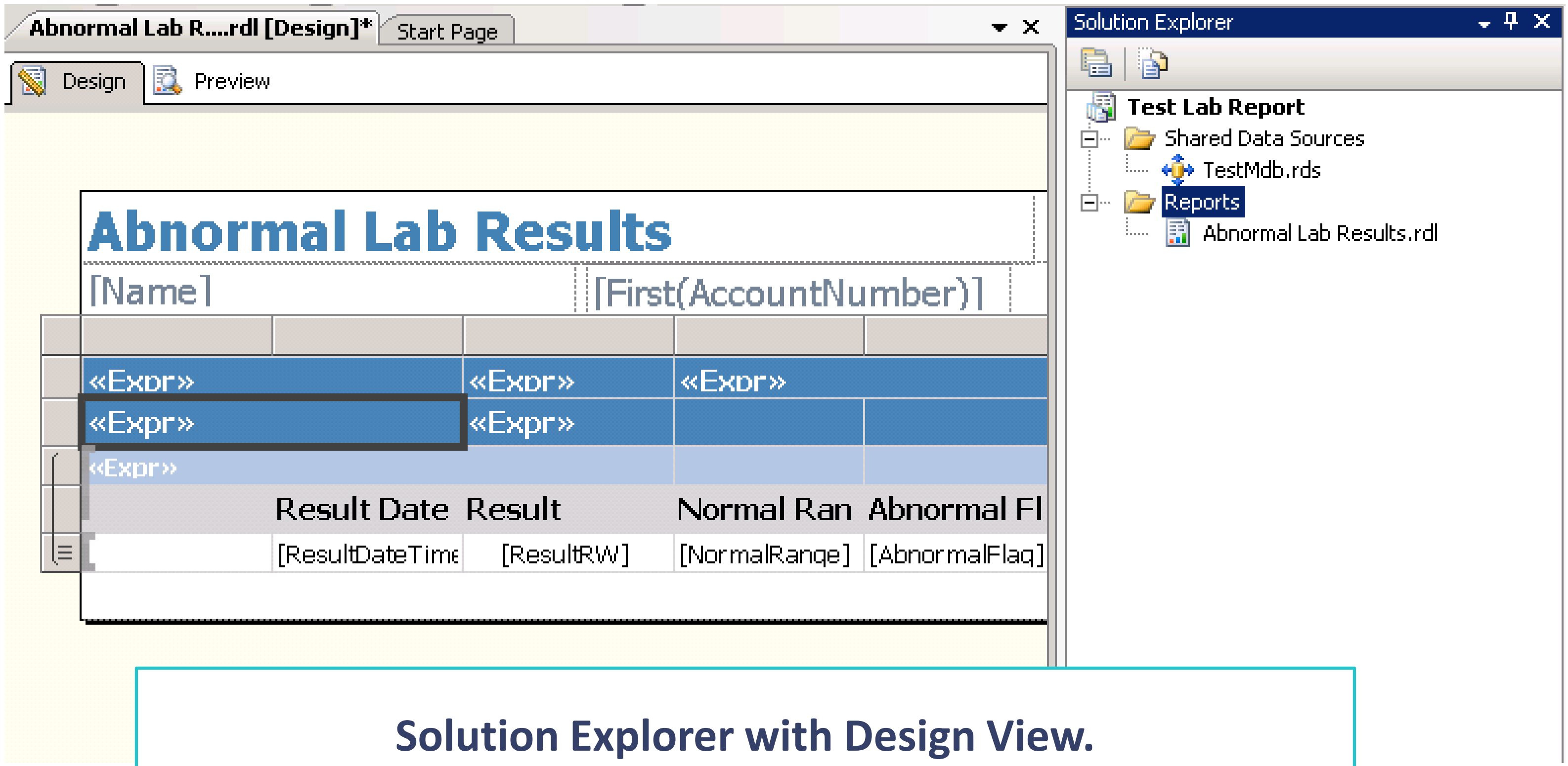

AcmewareTest
Execute
SQLQuery2.sql ...gerardo (80))*
SQLQuery1.sql ...jgerardo (87))*

```
ALTER PROC spMuseLabTests
AS


SELECT BV.VisitID,
    BV.AccountNumber,BV.Name,
    BV.FinancialClassID,
    BV.InpatientOrOutpatient,
    BV.Sex, BV.BirthDateTime,
    BV.PrimaryInsuranceID,
    TestMnemonic,
    TestName,
    ResultDateTime,
    NormalRange,
    ResultRW,
    AbnormalFlag,
CASE WHEN AbnormalFlag = '  *' THEN 'Other'
    WHEN AbnormalFlag like '%H%' THEN 'High'
    WHEN AbnormalFlag like '%L%' THEN 'Low'
    END AS AbnormalFlagText
FROM TestMdb.dbo.LabSpecimenTests LST
INNER JOIN TestMdb.dbo.BarVisits BV
ON LST.SourceID = BV.SourceID
AND LST.VisitID = BV.VisitID
WHERE DATEDIFF(MM,ResultDateTime, GETDATE()) < 3
ORDER BY 1,8,10
```

Example only

Lab results for the past 3 months .

SSRS Report Development



Abnormal Lab Results

[Name] [First(AccountNumber)]

«Expr» «Expr» «Expr»

«Expr» «Expr»

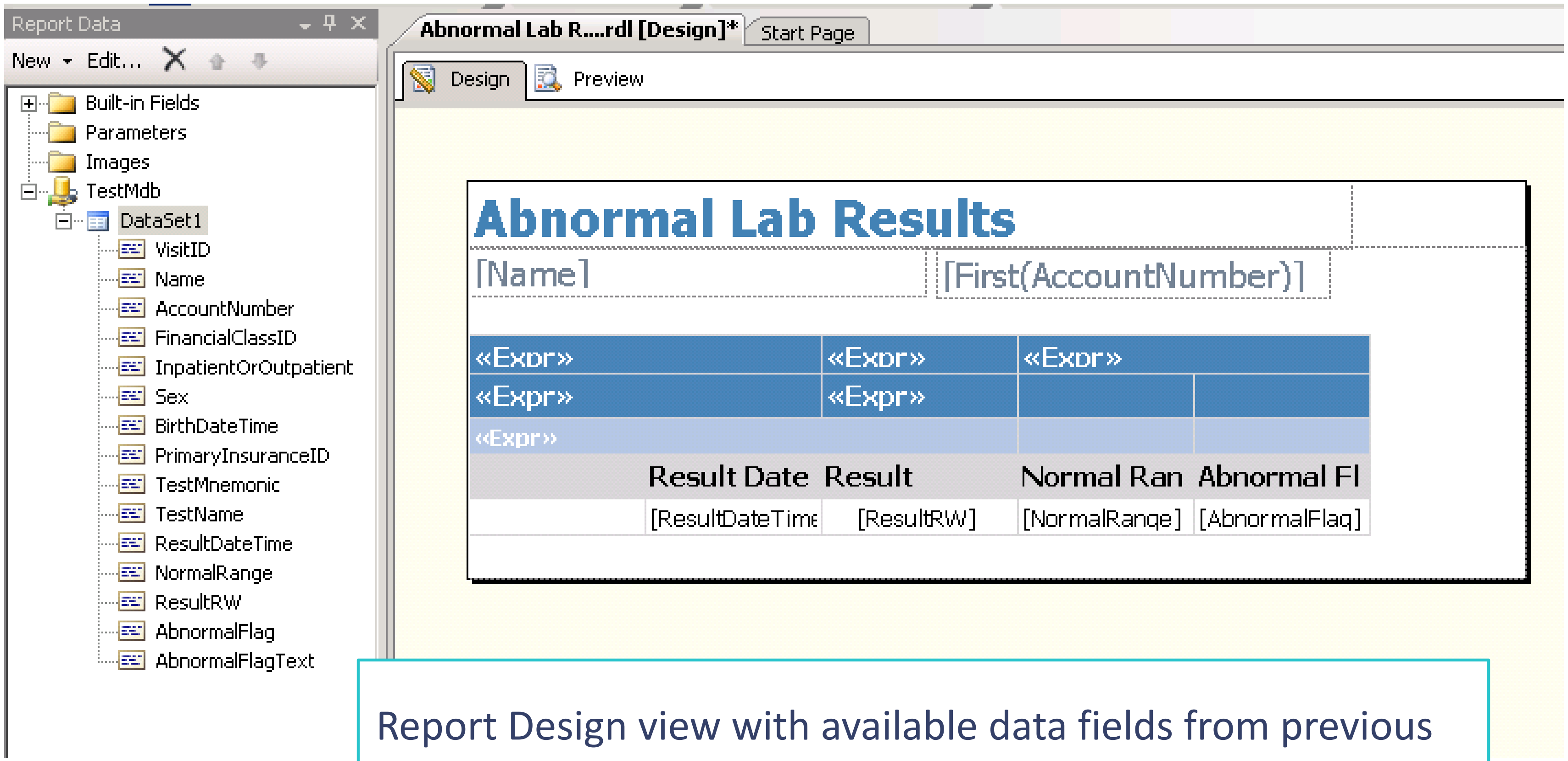
«Expr»

	Result Date	Result	Normal Ran	Abnormal Fl
	[ResultDateTime	[ResultRW]	[NormalRange]	[AbnormalFlag]

Solution Explorer with Design View.

Data Source – defines database connection
Reports– contains all developed reports

SSRS Development


Report Data

New Edit...

Built-in Fields
Parameters
Images
TestMdb
DataSet1
VisitID
Name
AccountNumber
FinancialClassID
InpatientOrOutpatient
Sex
BirthDateTime
PrimaryInsuranceID
TestMnemonic
TestName
ResultDateTime
NormalRange
ResultRW
AbnormalFlag
AbnormalFlagText

Abnormal Lab R....rdl [Design]* Start Page

Design Preview

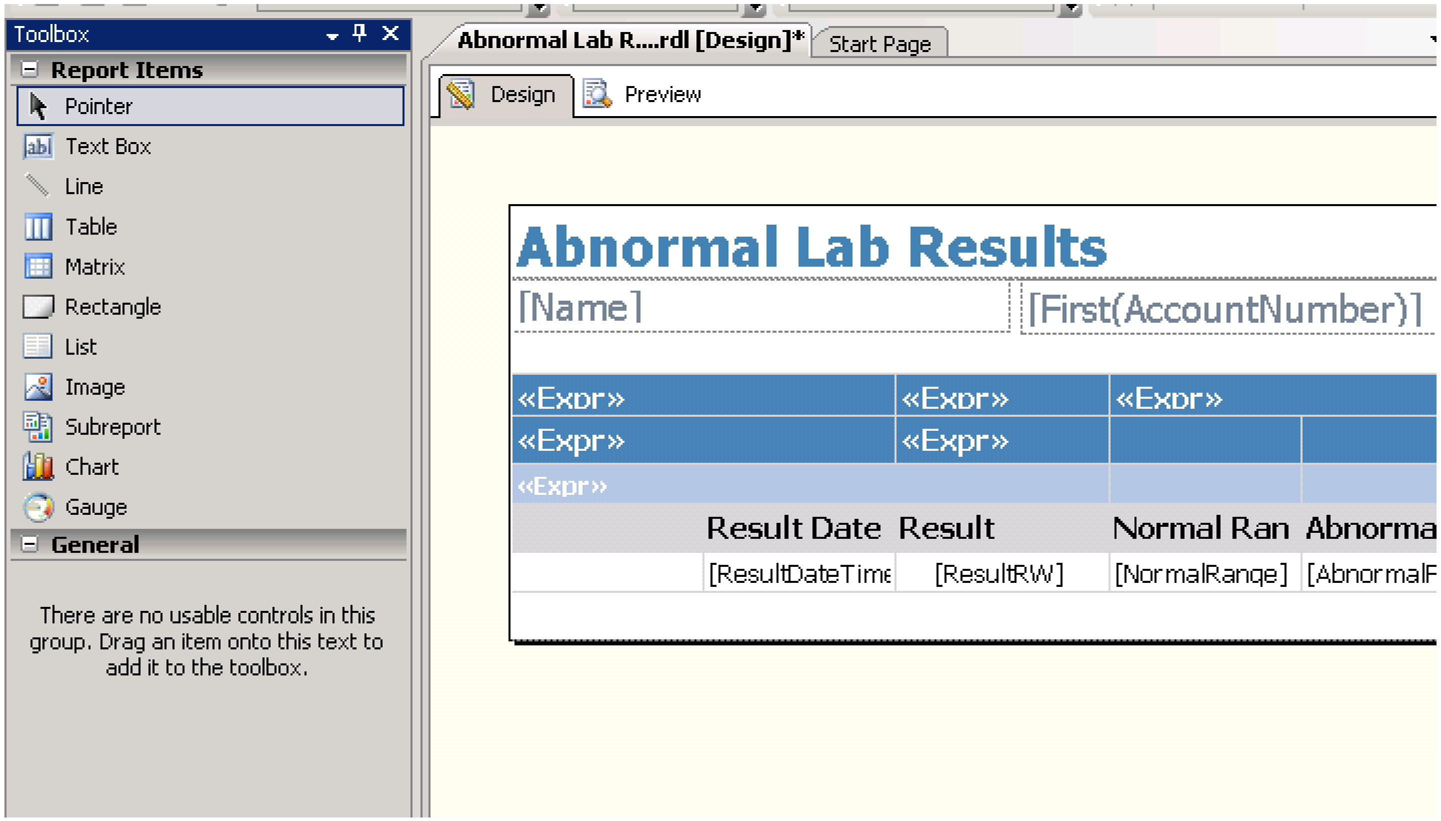
Abnormal Lab Results

[Name] [First(AccountNumber)]

«Expr»		«Expr»	«Expr»	
«Expr»		«Expr»		
«Expr»				
	Result Date	Result	Normal Ran	Abnormal Fl
	[ResultDateTime	[ResultRW]	[NormalRange]	[AbnormalFlag]

Report Design view with available data fields from previous stored procedure

SSRS Development



There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

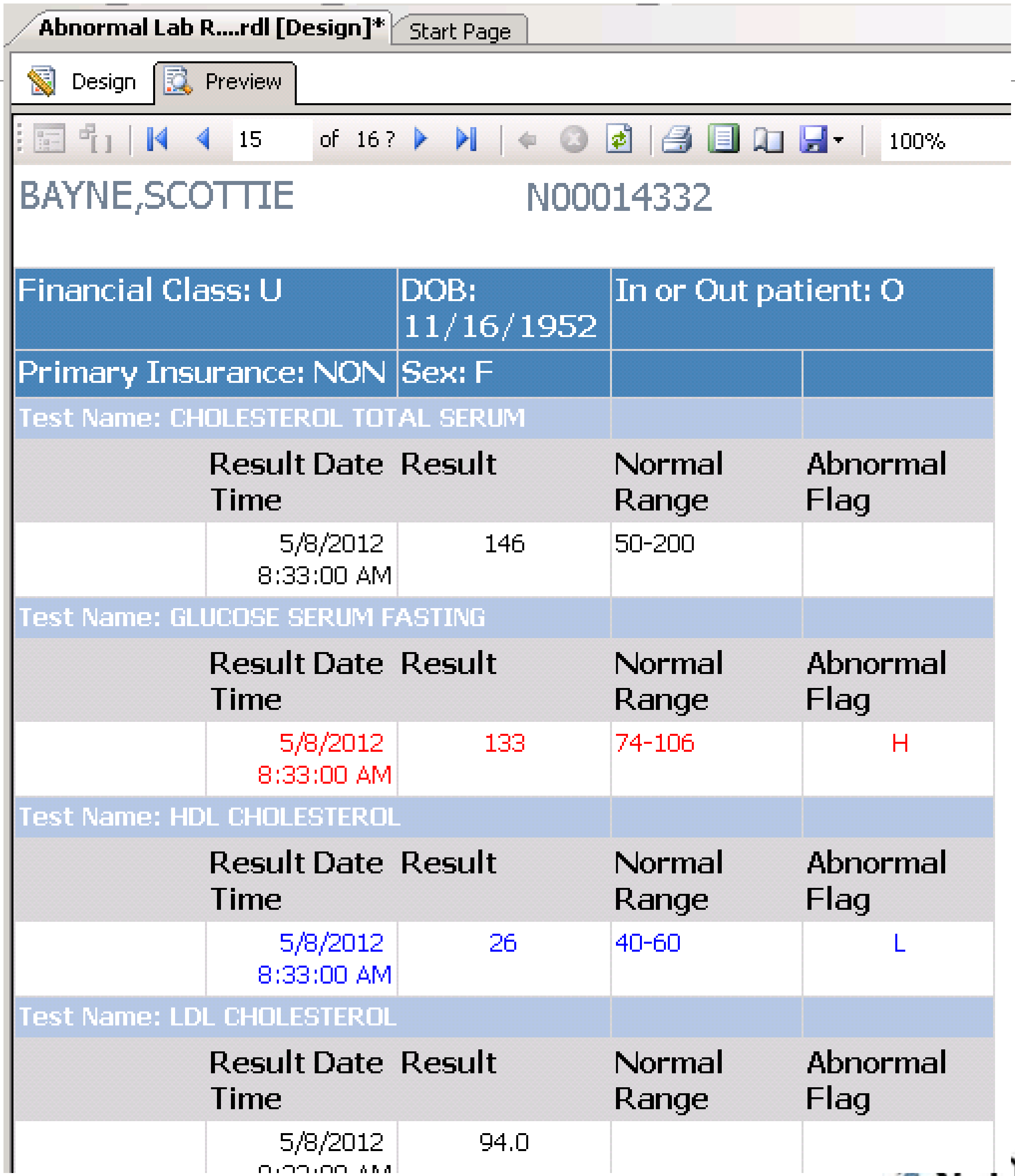
Abnormal Lab Results

[Name] [First(AccountNumber)]

«Expr»	«Expr»	«Expr»	
«Expr»	«Expr»		
«Expr»			
Result Date	Result	Normal Ran	Abnorma
[ResultDateTime	[ResultRW]	[NormalRange]	[AbnormalF

SSRS Deployed report

Report example grouped by patient and lab test with details regarding test results



BAYNE,SCOTTIE N00014332

Financial Class: U	DOB: 11/16/1952	In or Out patient: O	
Primary Insurance: NON	Sex: F		

Test Name: CHOLESTEROL TOTAL SERUM

Result Date Time	Result	Normal Range	Abnormal Flag
5/8/2012 8:33:00 AM	146	50-200	

Test Name: GLUCOSE SERUM FASTING

Result Date Time	Result	Normal Range	Abnormal Flag
5/8/2012 8:33:00 AM	133	74-106	H

Test Name: HDL CHOLESTEROL

Result Date Time	Result	Normal Range	Abnormal Flag
5/8/2012 8:33:00 AM	26	40-60	L

Test Name: LDL CHOLESTEROL

Result Date Time	Result	Normal Range	Abnormal Flag
5/8/2012	94.0		

Look for our MUSE sessions

- Tuesday, May 29
  - 702 - Custom BCA Dashboards with Visual Insight
  - 703 - The Alphabet Soup of Clinical Quality Measures Reporting and Reimbursement: 2018 Updates
  - 704 - Soup to Nuts - Data Repository 101
  - 802 - Report Designer Fundamentals
  - 804 - Soup to Nuts – Data Repository 102
- 1010 - Revenue Cycle Optimization: Tools and Strategies for Success
  Wednesday May 30 at 2:30 pm
- 1087 - HIE: Effective Integration and Interoperability
  Thursday May 31 at 1:45 pm
- 1104 - The DR Overnight DBA
  Thursday May 31 at 2:45 pm
- 1091 - Electronic Reporting: Quality Management Cycle Concepts that Achieve Reliable Results
  Friday June 1 at 9:00 am
- 1103 - The Report Request Lifecycle
  Friday June 1 at 10:00 am

Acmeware