

(local) Documentation

LCCHPDev

Server (local)
Author liam
Created 18 July 2015 13:01
File Path D:\DBSpace\DB\LCCHP_documentation-2015-07-18T13-01-53.pdf

Table of Contents

Table of Contents	2
[local]	10
[User databases]	12
[LCCHPDev Database]	13
[Tables]	15
[dbo].[AccessAgreement]	19
[dbo].[AccessAgreementNotes]	22
[dbo].[AccessPurpose]	24
[dbo].[ActionStatus]	26
[dbo].[Area]	28
[dbo].[BloodTestResults]	30
[dbo].[BloodTestResultsNotes]	36
[dbo].[CleanupStatus]	38
[dbo].[Condition]	40
[dbo].[ConstructionType]	42
[dbo].[ContactType]	44
[dbo].[Contractor]	46
[dbo].[ContractortoProperty]	48
[dbo].[ContractortoRemediation]	50
[dbo].[ContractortoRemediationActionPlan]	53
[dbo].[Country]	55
[dbo].[DataSource]	57
[dbo].[Daycare]	59
[dbo].[DaycarePrimaryContact]	61
[dbo].[DaycaretoProperty]	63
[dbo].[Employer]	65
[dbo].[EmployertoProperty]	67
[dbo].[EnvironmentalInvestigation]	69
[dbo].[ErrorLog]	72
[dbo].[Ethnicity]	74
[dbo].[Family]	76
[dbo].[FamilyNotes]	79
[dbo].[FamilytoPhoneNumber]	81
[dbo].[FamilytoProperty]	83
[dbo].[FileType]	86
[dbo].[Flag]	88
[dbo].[ForeignFood]	90

[dbo].[ForeignFoodtoCountry]	92
[dbo].[Frequency]	94
[dbo].[GiftCard]	96
[dbo].[HistoricContribution]	98
[dbo].[Hobby]	100
[dbo].[HomeRemedy]	102
[dbo].[HouseholdSourcesofLead]	104
[dbo].[InsuranceProvider]	106
[dbo].[Lab]	108
[dbo].[LabNotes]	110
[dbo].[Language]	112
[dbo].[LCCHPAttachments]	114
[dbo].[Medium]	115
[dbo].[MediumSampleResults]	118
[dbo].[MediumSampleResultsNotes]	122
[dbo].[Method]	124
[dbo].[Occupation]	126
[dbo].[OccupationNotes]	128
[dbo].[Person]	130
[dbo].[PersonHobbyNotes]	137
[dbo].[PersonNotes]	139
[dbo].[PersonReleaseNotes]	141
[dbo].[PersonStatus]	143
[dbo].[PersontoAccessAgreement]	145
[dbo].[PersontoDaycare]	147
[dbo].[PersontoEmployer]	149
[dbo].[PersontoEthnicity]	151
[dbo].[PersontoFamily]	153
[dbo].[PersontoForeignFood]	155
[dbo].[PersontoHobby]	157
[dbo].[PersontoHomeRemedy]	159
[dbo].[PersontoInsurance]	161
[dbo].[PersontoLanguage]	163
[dbo].[PersontoOccupation]	165
[dbo].[PersontoPerson]	167
[dbo].[PersontoPhoneNumber]	170
[dbo].[PersontoProperty]	172
[dbo].[PersonToTravelCountry]	175

[dbo].[PersonTravelNotes]	177
[dbo].[PhoneNumber]	179
[dbo].[PhoneNumberType]	182
[dbo].[Property]	184
[dbo].[PropertyLinkType]	189
[dbo].[PropertyNotes]	191
[dbo].[PropertySampleResults]	193
[dbo].[PropertySampleResultsNotes]	196
[dbo].[PropertytoCleanupStatus]	198
[dbo].[PropertytoHouseholdSourcesofLead]	200
[dbo].[PropertytoMedium]	202
[dbo].[Questionnaire]	204
[dbo].[QuestionnaireDataSource]	211
[dbo].[QuestionnaireNotes]	213
[dbo].[RelationshipType]	215
[dbo].[ReleaseStatus]	217
[dbo].[Remediation]	219
[dbo].[RemediationActionPlan]	222
[dbo].[RemediationNotes]	225
[dbo].[ReviewStatus]	227
[dbo].[SampleLevelCategory]	229
[dbo].[SamplePurpose]	231
[dbo].[SampleType]	233
[dbo].[Source]	235
[dbo].[TargetStatus]	236
[dbo].[TravelNotes]	238
[dbo].[Units]	240
Views	242
[dbo].[vAdults]	243
[dbo].[vMostRecentBloodTestResults]	245
[dbo].[vMostRecentQuestionnaires]	247
[dbo].[vNursingInfants]	250
[dbo].[vNursingMothers]	251
[dbo].[vPregnant]	252
Stored Procedures	253
[dbo].[DELETE_usp_InsertPersonToStatus]	257
[dbo].[DELETE_usp_SICountClients]	259
[dbo].[TransProc]	262

[dbo].[usp_InsertAccessAgreement]	263
[dbo].[usp_InsertAccessPurpose]	266
[dbo].[usp_InsertArea]	268
[dbo].[usp_InsertBloodTestResults]	270
[dbo].[usp_InsertBloodTestResultsNotes]	274
[dbo].[usp_InsertCleanupStatus]	276
[dbo].[usp_InsertConstructionType]	278
[dbo].[usp_InsertContractor]	280
[dbo].[usp_InsertContractortoProperty]	282
[dbo].[usp_InsertContractortoRemediation]	284
[dbo].[usp_InsertContractortoRemediationActionPlan]	286
[dbo].[usp_InsertCountry]	288
[dbo].[usp_InsertDaycare]	290
[dbo].[usp_InsertDaycarePrimaryContact]	292
[dbo].[usp_InsertDaycaretoProperty]	294
[dbo].[usp_InsertEmployer]	296
[dbo].[usp_InsertEmployertoProperty]	298
[dbo].[usp_InsertEnvironmentalInvestigation]	300
[dbo].[usp_InsertEthnicity]	303
[dbo].[usp_InsertFamily]	305
[dbo].[usp_InsertFamilyNotes]	308
[dbo].[usp_InsertFamilytoPhoneNumber]	310
[dbo].[usp_InsertFamilytoProperty]	314
[dbo].[usp_InsertForeignFood]	317
[dbo].[usp_InsertForeignFoodtoCountry]	319
[dbo].[usp_InsertGiftCard]	321
[dbo].[usp_InsertHobby]	323
[dbo].[usp_InsertHomeRemedies]	325
[dbo].[usp_InsertHouseholdSourcesofLead]	327
[dbo].[usp_InsertInsuranceProvider]	329
[dbo].[usp_InsertLab]	331
[dbo].[usp_InsertLabNotes]	333
[dbo].[usp_InsertLanguage]	335
[dbo].[usp_InsertMedium]	337
[dbo].[usp_InsertMediumSampleResults]	339
[dbo].[usp_InsertMediumSampleResultsNotes]	342
[dbo].[usp_InsertNewBloodLeadTestResultsWebScreen]	344
[dbo].[usp_InsertNewClientWebScreen]	348

[dbo].[usp_InsertNewFamilyWebScreen]	353
[dbo].[usp_InsertNewQuestionnaireWebScreen]	360
[dbo].[usp_InsertOccupation]	365
[dbo].[usp_InsertPerson]	367
[dbo].[usp_InsertPersonHobbyNotes]	371
[dbo].[usp_InsertPersonNotes]	373
[dbo].[usp_InsertPersonReleaseNotes]	375
[dbo].[usp_InsertPersontoAccessAgreement]	377
[dbo].[usp_InsertPersontoDaycare]	379
[dbo].[usp_InsertPersontoEmployer]	381
[dbo].[usp_InsertPersontoEthnicity]	383
[dbo].[usp_InsertPersontoFamily]	385
[dbo].[usp_InsertPersontoForeignFood]	387
[dbo].[usp_InsertPersontoHobby]	389
[dbo].[usp_InsertPersontoHomeRemedy]	391
[dbo].[usp_InsertPersontoInsurance]	393
[dbo].[usp_InsertPersontoLanguage]	395
[dbo].[usp_InsertPersontoOccupation]	398
[dbo].[usp_InsertPersontoPerson]	400
[dbo].[usp_InsertPersontoPhoneNumber]	403
[dbo].[usp_InsertPersontoProperty]	405
[dbo].[usp_InsertPersontoTravelCountry]	407
[dbo].[usp_InsertPersonTravelNotes]	409
[dbo].[usp_InsertPhoneNumber]	411
[dbo].[usp_InsertPhoneNumberType]	414
[dbo].[usp_InsertProperty]	416
[dbo].[usp_InsertPropertyNotes]	420
[dbo].[usp_InsertPropertySampleResults]	422
[dbo].[usp_InsertPropertySampleResultsNotes]	425
[dbo].[usp_InsertPropertytoCleanupStatus]	427
[dbo].[usp_InsertPropertytoHouseholdSourcesofLead]	429
[dbo].[usp_InsertPropertytoMedium]	431
[dbo].[usp_InsertQuestionnaire]	433
[dbo].[usp_InsertQuestionnaireNotes]	437
[dbo].[usp_InsertRemediation]	439
[dbo].[usp_InsertRemediationActionPlan]	442
[dbo].[usp_InsertRemediationNotes]	445
[dbo].[usp_InsertSampleLevelCategory]	447

[dbo].[usp_InsertSampleType]	449
[dbo].[usp_InsertStatus]	451
[dbo].[usp_InsertTravelNotes]	453
[dbo].[usp_SLAllBloodTestResults]	455
[dbo].[usp_SLAllBloodTestResults2]	458
[dbo].[usp_SLAllBloodTestResultsMetaData]	461
[dbo].[usp_SIChildStatus]	463
[dbo].[usp_SIClientFollowUp]	465
[dbo].[usp_SIColumnDetails]	468
[dbo].[usp_SICountAdults]	470
[dbo].[usp_SICountBloodLeadLevels]	473
[dbo].[usp_SICountBloodTests]	476
[dbo].[usp_SICountClients]	479
[dbo].[usp_SICountFamilyMembers]	482
[dbo].[usp_SICountHomeVisitSoilSample]	485
[dbo].[usp_SICountNewClients]	488
[dbo].[usp_SICountNewPeople]	491
[dbo].[usp_SICountNursingInfants]	494
[dbo].[usp_SICountNursingMothers]	497
[dbo].[usp_SICountPeople]	500
[dbo].[usp_SICountPeopleByAge]	502
[dbo].[usp_SICountPeopleByAgeGroup]	504
[dbo].[usp_SICountPeopleByLastName]	506
[dbo].[usp_SICountPregnantWomen]	508
[dbo].[usp_SIDaycare]	511
[dbo].[usp_SIEditBloodTestResultsWebScreenInformation]	513
[dbo].[usp_SIEditClientInfoWebScreenInformation]	516
[dbo].[usp_SIEditFamilyWebScreenInformation]	519
[dbo].[usp_SIEditPropertyWebScreenInformation]	522
[dbo].[usp_SIEditQuestionnaireWebScreenInformation]	525
[dbo].[usp_SIFamilyMembers]	528
[dbo].[usp_SIFamilyNameToProperty]	531
[dbo].[usp_SIHobby]	533
[dbo].[usp_SIInsertedData]	535
[dbo].[usp_SLInsertedDataSimplified]	539
[dbo].[usp_SILabName]	542
[dbo].[usp_SLLListAllFamilyMembers]	544
[dbo].[usp_SIListClientsByCreatedate]	547

[dbo].[usp_SIListClientsByModifiedDate]	550
[dbo].[usp_SIListFamilies]	553
[dbo].[usp_SIListFamilyMembers]	555
[dbo].[usp_SIListNursingWomenbyCreateDateRange]	558
[dbo].[usp_SIListPeoplebyCreateDateRange]	561
[dbo].[usp_SLLListPotentialDuplicatePeople]	564
[dbo].[usp_SLLListPotentialDuplicateProperties]	566
[dbo].[usp_SIListPregnantWomenbyCreateDateRange]	568
[dbo].[usp_SLMostRecentBloodTestResults]	571
[dbo].[usp_SIPersonNotes]	574
[dbo].[usp_SIPersonsToEthnicity]	577
[dbo].[usp_SIPersonsToLanguage]	580
[dbo].[usp_SIRelationshipTypes]	583
[dbo].[usp_SIStatus]	585
[dbo].[usp_SISummaryReport]	587
[dbo].[usp_SISummaryReport_MetaData]	597
[dbo].[usp_SITargetSampleType]	600
[dbo].[usp_upBloodTestResults]	602
[dbo].[usp_upBloodTestResultsWebScreen]	606
[dbo].[usp_upClientFlag]	610
[dbo].[usp_upClientWebScreen]	612
[dbo].[usp_upFamily]	618
[dbo].[usp_upFamilytoProperty]	622
[dbo].[usp_upFamilyWebScreen]	625
[dbo].[usp_upOccupation]	632
[dbo].[usp_upPerson]	635
[dbo].[usp_upProperty]	642
[dbo].[usp_upQuestionnaire]	648
[dbo].[usp_upQuestionnaireWebScreen]	654
[dbo].[uspLogError]	658
[dbo].[uspPrintError]	663
 Scalar-valued Functions	667
 [dbo].[RemoveSpecialChars]	668
 [dbo].[udf_CalculateAge]	670
 [dbo].[udf_DateInThePast]	672
 [dbo].[udf_DoesPropertyExist]	674
 [dbo].[udf_SIFamilyPhoneNumber]	676
 Users	678

👤 appUser	679
👤 WIN-1M8NQQ69OEH\SQLMaintenenace	680
👤 Database Roles	681
👤 db_accessadmin	681
👤 db_backupoperator	681
👤 db_datareader	682
👤 db_datawriter	682
👤 db_ddladmin	683
👤 db_denydatareader	683
👤 db_denydatawriter	684
👤 db_owner	684
👤 db_securityadmin	684
👤 public	685

 (local)
--

Databases(1)

-  LCCHPDev

Server Properties

Property	Value
Product	Microsoft SQL Server
Version	11.0.5058.0
Language	English (United States)
Platform	NT x64
Edition	Express Edition (64-bit)
Processors	2
OS Version	6.2 (9200)
Physical Memory	4096
Is Clustered	False
Root Directory	C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL
Collation	SQL_Latin1_General_CI_AS

Server Settings

Property	Value
Default data file path	D:\MSSQL\Data\
Default backup file path	Y:\Backup\WIN-1M8NQQ69OEH
Default log file path	D:\MSSQL\Log\
Recovery Interval (minutes)	0
Default index fill factor	0
Default backup media retention	0

Advanced Server Settings

Property	Value
Full text upgrade option	2
Locks	0
Nested triggers enabled	True
Allow triggers to fire others	True
Default language	English
Network packet size	4096

Project> (local)

Default fulltext language LCID	1033
Two-digit year cutoff	2049
Remote login timeout	10
Cursor threshold	-1
Max text replication size	65536
Parallelism cost threshold	25
Scan for startup procs	False
Transform noise words	False
Blocked process threshold	0
Filestream access level	2
Optimize for ad hoc workloads	True

Project> (local)> User databases

 **User databases**

Databases(1)

-  LCCHPDev

LCCHPDev Database

Database Properties

Property	Value
SQL Server Version	SQL Server 2012
Compatibility Level	SQL Server 2012
Database Encryption Enabled	False
Last backup time	07/18/2015
Last log backup time	07/18/2015
Creation date	Jan 2 2015
Users	6
Database size	45.00 MB
Unallocated space	5.47 MB

Database Options

Property	Value
Compatibility Level	110
Database collation	SQL_Latin1_General_CI_AS
Restrict access	MULTI_USER
Is read-only	False
Auto close	False
Auto shrink	False
Database status	ONLINE
In standby	False
Cleanly shutdown	False
Supplemental logging enabled	False
Snapshot isolation state	OFF
Read committed snapshot on	False
Recovery model	FULL
Page verify option	CHECKSUM
Auto create statistics	True
Auto update statistics	True
Auto update statistics asynchronously	False
ANSI NULL default	False
ANSI NULL enabled	False
ANSI padding enabled	False
ANSI warnings enabled	False

Arithmetic abort enabled	False
Concatenating NULL yields NULL	False
Numeric roundabort enabled	False
Quoted Identifier On	False
Recursive triggers enabled	False
Close cursors on commit	False
Local cursors by default	False
Fulltext enabled	True
Trustworthy	False
Database chaining	False
Forced parameterization	False
Master key encrypted by server	False
Published	False
Subscribed	False
Merge published	False
Is distribution database	False
Sync with backup	False
Service broker GUID	5e0a2947-6106-454c-a34c-8e1821adb0ff
Service broker enabled	False
Log reuse wait	LOG_BACKUP
Date correlation	False
CDC enabled	False
Encrypted	False
Honor broker priority	False
Default language	English
Default fulltext language LCID	1033
Nested triggers enabled	True
Transform noise words	False
Two-digit year cutoff	2049
Containment	NONE
Target recovery time	0
Database owner	WIN-1M8NQQ69OEH\liam

Files

Name	Type	File Group	Size	File Name
LCCHP	Data		8.00 MB	D:\MSSQL\Data\LCCHPDev.mdf
LCCHP_log	Log		25.00 MB	D:\MSSQL\Log\LCCHPDev_log.ldf
LCCHP_UData	Data	UData	12.00 MB	D:\MSSQL\Data\LCCHPDev_UData.ndf
LCCHPAttachments	Filestream			D:\MSSQL\Filestream\LCCHPAttachmentsDev

 Tables

Objects

Name
dbo.AccessAgreement collection of access agreements
dbo.AccessAgreementNotes linking table for access agreement and access agreement notes
dbo.AccessPurpose collection of purposes for access requests/agreements
dbo.ActionStatus Collection of potential status for Action
dbo.Area collection of areas and basic information
dbo.BloodTestResults Collection of blood test result values and categorization
dbo.BloodTestResultsNotes linking table for access agreement and access agreement notes
dbo.CleanupStatus collection of clean up status
dbo.Condition Collection of potential status for Action
dbo.ConstructionType collection of construction types
dbo.ContactType Collection of contact types
dbo.Contractor
dbo.ContractortoProperty linking table for contractor and occupied properties
dbo.ContractortoRemediation linking table for contractors and remediations
dbo.ContractortoRemediationActionPlan linking table for contractor and sampling plan
dbo.Country collection of countries
dbo.DataSource Collection of contact types
dbo.Daycare collection of daycare facilities
dbo.DaycarePrimaryContact linking table for daycare and person - identifying contact person
dbo.DaycaretoProperty linking table for daycare and property
dbo.Employer collection of employers
dbo.EmployertoProperty

linking table for employer and property
dbo.EnvironmentalInvestigation
dbo.ErrorLog
dbo.Ethnicity collection of ethnicities
dbo.Family collection of families
dbo.FamilyNotes table for Family notes
dbo.FamilytoPhoneNumber linking table for Family and phonenumber
dbo.FamilytoProperty linking table for Family and property - indicating when a Family occupied a property
dbo.FileType
dbo.Flag Collection of flag information
dbo.ForeignFood collection of various foreign foods
dbo.ForeignFoodtoCountry foreign food and country linking table
dbo.Frequency Collection of frequencies
dbo.GiftCard collection of gift certificate objects
dbo.HistoricContribution Collection of historic contribution classifications
dbo.Hobby collection of hobbies
dbo.HomeRemedy collection of home remedies
dbo.HouseholdSourcesofLead household items that may contribute to EBL
dbo.InsuranceProvider collection of insurance companies
dbo.Lab collection of lab names and basic attributes
dbo.LabNotes linking table for access agreement and access agreement notes
dbo.Language collection of spoken languages
dbo.LCCHPAttachments
dbo.Medium collection of mediums that are tested
dbo.MediumSampleResults collection of test results for various mediums
dbo.MediumSampleResultsNotes linking table for access agreement and access agreement notes

dbo.Method
Collection of method classifications
dbo.Occupation
collection of occupation objects
dbo.OccupationNotes
table for Occupation notes
dbo.Person
collection of people and basic attributes
dbo.PersonHobbyNotes
table for person hobby notes
dbo.PersonNotes
table for person notes
dbo.PersonReleaseNotes
table for person release notes
dbo.PersonStatus
Collection of potential status for person
dbo.PersontoAccessAgreement
linking table for person and access agreement
dbo.PersontoDaycare
linking table for person and daycare for people attending daycare
dbo.PersontoEmployer
linking table for person and employer
dbo.PersontoEthnicity
linking table for person and ethnicity
dbo.PersontoFamily
linking table for person and family tables
dbo.PersontoForeignFood
linking table for person and foreign food (many to many)
dbo.PersontoHobby
linking table for person and hobby
dbo.PersontoHomeRemedy
linking table for person and home remedy
dbo.PersontoInsurance
linking table for person and insurance
dbo.PersontoLanguage
linking table for person and language
dbo.PersontoOccupation
linking table for person and occupation
dbo.PersontoPerson
collection of relationships between people
dbo.PersontoPhoneNumber
linking table for person and phonenumbers
dbo.PersontoProperty
linking table for person and property - indicating when a person occupied a property
dbo.PersonToTravelCountry
linking table for person and country traveled to
dbo.PersonTravelNotes
table for person Travel notes
dbo.PhoneNumber
collection of phone number objects
dbo.PhoneNumberType

dbo.Property collection of properties and basic attributes
dbo.PropertyLinkType Collection of property link types
dbo.PropertyNotes linking table for access agreement and access agreement notes
dbo.PropertySampleResults collection of property test results
dbo.PropertySampleResultsNotes linking table for access agreement and access agreement notes
dbo.PropertytoCleanupStatus linking table for property and cleanup status
dbo.PropertytoHouseholdSourcesofLead linking table for property and household sources of lead
dbo.PropertytoMedium linking table for property and media
dbo.Questionnaire collection of questionnaire questions and answers, typically only completed by flagged patients
dbo.QuestionnaireDataSource source of the data (Environmental group or Blood Lead)
dbo.QuestionnaireNotes linking table for access agreement and access agreement notes
dbo.RelationshipType collection of RelationshipType names and basic attributes
dbo.ReleaseStatus Collection of Release Status
dbo.Remediation collection of remediation data
dbo.RemediationActionPlan collection of sampling plans
dbo.RemediationNotes table for remediation notes
dbo.ReviewStatus Collection of potential status for Review
dbo.SampleLevelCategory collection of sample level categorizations
dbo.SamplePurpose Collection of sample purposes
dbo.SampleType collection of sample types
dbo.Source
dbo.TargetStatus collection of status objects
dbo.TravelNotes Collection of family and travel notes
dbo.Units

[dbo].[AccessAgreement]

MS_Description

collection of access agreements

Properties

Property	Value
File Group	UData
Row Count (~)	5
Created	7:28:57 PM Friday, December 26, 2014
Last Modified	10:36:24 AM Saturday, February 14, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	AccessAgreementID	int	4	False	1 - 1	
 D	AccessPurposeID id of the access purpose	int	4	True		
	AccessAgreementFile	varbinary(max)	max	True		
 E	PropertyID	int	4	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_AccessAgreement	AccessAgreementID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateAccessAgreement	True	True	After Update

Foreign Keys

Name	Columns
FK_AccessAgreement_AccessPurpose	AccessPurposeID->[dbo].[AccessPurpose].[AccessPurposeID]
FK_AccessAgreement_Property	PropertyID->[dbo].[Property].[PropertyID]

SQL Script

```

CREATE TABLE [dbo].[AccessAgreement]
(
    [AccessAgreementID] [int] NOT NULL IDENTITY(1, 1),
    [AccessPurposeID] [int] NULL,
    [AccessAgreementFile] [varbinary] (max) NULL,
    [PropertyID] [int] NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_AccessAgreement_CreatedDate] DEFAULT
    (getdate())
) ON [UData] TEXTIMAGE_ON [UData]
GO
create trigger [dbo].[trUpdateAccessAgreement] on [dbo].[AccessAgreement] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update AccessAgreement set ModifiedDate = getdate()
    where AccessAgreementID in (select AccessAgreementID from inserted)

end
GO
ALTER TABLE [dbo].[AccessAgreement] ADD CONSTRAINT [PK_AccessAgreement] PRIMARY KEY
CLUSTERED ([AccessAgreementID]) ON [UData]
GO
ALTER TABLE [dbo].[AccessAgreement] ADD CONSTRAINT [FK_AccessAgreement_AccessPurpose]
FOREIGN KEY ([AccessPurposeID]) REFERENCES [dbo].[AccessPurpose] ([AccessPurposeID])
GO
ALTER TABLE [dbo].[AccessAgreement] ADD CONSTRAINT [FK_AccessAgreement_Property]
FOREIGN KEY ([PropertyID]) REFERENCES [dbo].[Property] ([PropertyID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of access agreements',
'SCHEMA', N'dbo', 'TABLE', N'AccessAgreement', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the access purpose', 'SCHEMA',
N'dbo', 'TABLE', N'AccessAgreement', 'COLUMN', N'AccessPurposeID'
GO

```

Uses

[dbo].[AccessPurpose]
 [dbo].[Property]

Used By

[dbo].[AccessAgreementNotes]
[dbo].[PersonstoAccessAgreement]
[dbo].[usp_InsertAccessAgreement]

[dbo].[AccessAgreementNotes]

MS_Description

linking table for access agreement and access agreement notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	4
Created	10:36:24 AM Saturday, February 14, 2015
Last Modified	10:36:24 AM Saturday, February 14, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	AccessAgreementNotesID	int	4	False	1 - 1	
 D	AccessAgreementID	int	4	False		
	CreatedDate date the notes where added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	True		

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_AccessAgreementNotes	AccessAgreement-NotesID	True	UData

Foreign Keys

Name	Columns
FK_AccessAgreementNotes_AccessAgreement	AccessAgreementID->[dbo].[AccessAgreement].[AccessAgreementID]

SQL Script

```
CREATE TABLE [dbo] . [AccessAgreementNotes]
```

Author: liam

```
(  
[AccessAgreementNotesID] [int] NOT NULL IDENTITY(1, 1),  
[AccessAgreementID] [int] NOT NULL,  
[CreatedDate] [datetime] NULL CONSTRAINT [DF_AccessAgreementNotes_CreatedDate] DEFAULT  
(getdate()),  
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NULL  
) ON [UData]  
GO  
ALTER TABLE [dbo].[AccessAgreementNotes] ADD CONSTRAINT [PK_AccessAgreementNotes]  
PRIMARY KEY CLUSTERED ([AccessAgreementNotesID]) ON [UData]  
GO  
ALTER TABLE [dbo].[AccessAgreementNotes] ADD CONSTRAINT [FK_AccessAgreementNotes_Access-  
Agreement] FOREIGN KEY ([AccessAgreementID]) REFERENCES [dbo].[AccessAgreement]  
([AccessAgreementID])  
GO  
EXEC sp_addextendedproperty N'MS_Description', N'linking table for access agreement and  
access agreement notes', 'SCHEMA', N'dbo', 'TABLE', N'AccessAgreementNotes', NULL, NULL  
GO  
EXEC sp_addextendedproperty N'MS_Description', N'date the notes where added', 'SCHEMA',  
N'dbo', 'TABLE', N'AccessAgreementNotes', 'COLUMN', N'CreatedDate'  
GO
```

Uses

[dbo].[AccessAgreement]

Used By

[dbo].[usp_InsertAccessAgreement]

[dbo].[AccessPurpose]

MS_Description

collection of purposes for access requests/agreements

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	6
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	4:55:06 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	AccessPurposeID	int	4	False	1 - 1	
	AccessPurposeName friendly name for the access purpose	varchar(50)	50	True		
	AccessPurposeDescription a description of the access purpose	varchar(253)	253	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_AccessPurpose	AccessPurposeID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateAccessPurpose	True	True	After Update

SQL Script

```

CREATE TABLE [dbo].[AccessPurpose]
(
[AccessPurposeID] [int] NOT NULL IDENTITY(1, 1),
[AccessPurposeName] [varchar](50) COLLATE SQL_Latin1_General_CI_AS NULL,
[AccessPurposeDescription] [varchar](253) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_AccessPurpose_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
create trigger [dbo].[trUpdateAccessPurpose] on [dbo].[AccessPurpose] AFTER UPDATE
as
begin
if @@rowcount = 0
return
if not update(ModifiedDate) update AccessPurpose set ModifiedDate = getdate()
where AccessPurposeID in (select AccessPurposeID from inserted)

end
GO
ALTER TABLE [dbo].[AccessPurpose] ADD CONSTRAINT [PK_AccessPurpose] PRIMARY KEY
CLUSTERED ([AccessPurposeID]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of purposes for access
requests/agreements', 'SCHEMA', N'dbo', 'TABLE', N'AccessPurpose', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'a description of the access purpose',
'SCHEMA', N'dbo', 'TABLE', N'AccessPurpose', 'COLUMN', N'AccessPurposeDescription'
GO
EXEC sp_addextendedproperty N'MS_Description', N'friendly name for the access purpose',
'SCHEMA', N'dbo', 'TABLE', N'AccessPurpose', 'COLUMN', N'AccessPurposeName'
GO

```

Used By

[dbo].[AccessAgreement]
[dbo].[usp_InsertAccessPurpose]

[dbo].[ActionStatus]	

MS_Description

Collection of potential status for Action

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	0
Created	2:12:35 PM Saturday, April 11, 2015
Last Modified	2:12:35 PM Saturday, April 11, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	ActionStatusID	tinyint	1	False	1 - 1	
	ActionStatusDescription Detailed description of the Action status	varchar(253)	253	True		
	ActionStatusName status for the Action	varchar(50)	50	True		
	HistoricActionStatusID historic status from access database	char(1)	1	True		
	ModifiedDate last modified date for the record	datetime	8	True		
	CreatedDate date the record was created	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_ActionStatus	ActionStatusID	True	UData

SQL Script

```
CREATE TABLE [dbo].[ActionStatus]
(
[ActionStatusID] [tinyint] NOT NULL IDENTITY(1, 1),
```

Author: liam

Project> (local)> User databases> LCCHPDev> Tables> dbo.ActionStatus

```
[ActionStatusDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[ActionStatusName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[HistoricActionStatusID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_ActionStatus_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[ActionStatus] ADD CONSTRAINT [PK_ActionStatus] PRIMARY KEY CLUSTERED
([ActionStatusID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Collection of potential status for
Action', 'SCHEMA', N'dbo', 'TABLE', N'ActionStatus', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Detailed description of the Action
status', 'SCHEMA', N'dbo', 'TABLE', N'ActionStatus', 'COLUMN', N'ActionStatus-
Description'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'status for the Action', 'SCHEMA',
N'dbo', 'TABLE', N'ActionStatus', 'COLUMN', N'ActionStatusName'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the record was created',
'SCHEMA', N'dbo', 'TABLE', N'ActionStatus', 'COLUMN', N'CreatedDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'historic status from access database',
'SCHEMA', N'dbo', 'TABLE', N'ActionStatus', 'COLUMN', N'HistoricActionStatusID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'last modified date for the record',
'SCHEMA', N'dbo', 'TABLE', N'ActionStatus', 'COLUMN', N'ModifiedDate'
GO
```

[dbo].[Area]

MS_Description

collection of areas and basic information

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	24
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	9:07:29 PM Thursday, April 9, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	AreaID unique identifier of the area	int	4	False	1 - 1	
	AreaDescription friendly description/name of the area	varchar(253)	253	True		
	HistoricAreaID	varchar(50)	50	False		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_Area	AreaID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateArea	True	True	After Update

SQL Script

```

CREATE TABLE [dbo].[Area]
(
[AreaID] [int] NOT NULL IDENTITY(1, 1),
[AreaDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[HistoricAreaID] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Area_CreatedDate] DEFAULT (getdate())
) ON [UData]
GO
create trigger [dbo].[trUpdateArea] on [dbo].[Area] AFTER UPDATE
as
begin
if @@rowcount = 0
    return
    if not update(ModifiedDate) update Area set ModifiedDate = getdate() where AreaID
in (select AreaID from inserted)

end
GO
ALTER TABLE [dbo].[Area] ADD CONSTRAINT [PK_Area] PRIMARY KEY CLUSTERED ([AreaID]) ON
[UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of areas and basic
information', 'SCHEMA', N'dbo', 'TABLE', N'Area', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'friendly description/name of the
area', 'SCHEMA', N'dbo', 'TABLE', N'Area', 'COLUMN', N'AreaDescription'
GO
EXEC sp_addextendedproperty N'MS_Description', N'unique identifier of the area',
'SCHEMA', N'dbo', 'TABLE', N'Area', 'COLUMN', N'AreaID'
GO

```

Used By

[dbo].[Property]
[dbo].[usp_InsertArea]

[dbo].[BloodTestResults]

MS_Description

Collection of blood test result values and categorization

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	13151
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	7:36:15 PM Thursday, June 11, 2015

Columns

Key	Name	Data Type	Computed	Max Length (Bytes)	Allow Nulls	Identity	Default
	BloodTestResultsID unique identifier for the blood test results object	int		4	False	1 - 1	
	isBaseline 0 = no; 1 = yes	bit		1	False		((0))
	PersonID	int		4	True		
	SampleDate date the sample was taken	date		3	False		(getdate())
	LabSubmissionDate date the sample was submitted to the lab	date		3	True		
	LeadValue	numeric(4,1)		5	True		
	LeadValueCategoryID id of the associated lead value categorization	tinyint		1	True		
	HemoglobinValue	numeric(4,1)		5	True		
	HemoglobinValueCategoryID id of the associated hemoglobin value categorization	tinyint		1	True		
	HematocritValueCategoryID id of the associated hematocrit value categorization	tinyint		1	True		

	LabID id of the lab to which the samples were submitted	int		4	True		
	BloodTestCosts cost of the blood tests	money		8	True		
	SampleTypeID id of the type of sample; i.e. venus, capo, soil, water, nitton analyzer . . .	tinyint		1	True		
	TakenAfterProperty-RemediationCompleted 0 - No, 1 - yes; was the blood sample taken after property remediation was completed.	bit		1	True		((0))
	ModifiedDate	datetime		8	True		
	CreatedDate	datetime		8	True		(getdate())
	HematocritValue	numeric(6,1)	True	5	True		
	ExcludeResult	bit		1	True		
	HistoricBloodTestResultsID Historic bloodpbresults id from access database	int		4	True		
	HistoricLabResultsID historic lab results id from access database	varchar(10)		10	True		
	ClientStatusID	smallint		2	True		

Computed columns

Name	Column definition
HematocritValue	([hemoglobinValue]*(3))

Indexes

Key	Name	Columns	Unique	Fill Factor	File Group
	PK_BloodTestResults	BloodTest-ResultsID	True		UData
	IDX_BloodTestResultsSampleDateLeadValue	BloodTest-ResultsID, ClientStatus-ID, Lab-Submission-Date, Person-ID, Sample-Date, Lead-Value		90	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateBloodTestResults	True	True	After Update

Check Constraints

Name	On Column	Constraint
ck_BloodTestResults_SampleDate	SampleDate	([dbo].[udf_DateInThePast]([SampleDate])=(1))

Foreign Keys

Name	Columns
FK_BloodTestResults_TargetStatus	ClientStatusID->[dbo].[TargetStatus].[StatusID]
FK_BloodTestResults_HematocritLevelCategory	HematocritValueCategoryID->[dbo].[SampleLevel-Category].[SampleLevelCategoryID]
FK_BloodTestResults_HemoglobinLevelCategory	HemoglobinValueCategoryID->[dbo].[SampleLevel-Category].[SampleLevelCategoryID]
FK_BloodTestResults_Lab	LabID->[dbo].[Lab].[LabID]
FK_BloodTestResults_LeadLevelCategory	LeadValueCategoryID->[dbo].[SampleLevel-Category].[SampleLevelCategoryID]
FK_BloodTestResults_Person	PersonID->[dbo].[Person].[PersonID]
FK_BloodTestResults_SampleType	SampleTypeID->[dbo].[SampleType].[SampleTypeID]

SQL Script

```

CREATE TABLE [dbo].[BloodTestResults]
(
[BloodTestResultsID] [int] NOT NULL IDENTITY(1, 1),
[isBaseline] [bit] NOT NULL CONSTRAINT [DF_BloodTestResults_isBaseline] DEFAULT ((0)),
[PersonID] [int] NULL,
[SampleDate] [date] NOT NULL CONSTRAINT [DF_BloodTestResults_SampleDate] DEFAULT (getdate()),
[LabSubmissionDate] [date] NULL,
[LeadValue] [numeric] (4, 1) NULL,
[LeadValueCategoryID] [tinyint] NULL,
[HemoglobinValue] [numeric] (4, 1) NULL,
[HemoglobinValueCategoryID] [tinyint] NULL,
[HematocritValueCategoryID] [tinyint] NULL,
[LabID] [int] NULL,
[BloodTestCosts] [money] NULL,
[SampleTypeID] [tinyint] NULL,
[TakenAfterPropertyRemediationCompleted] [bit] NULL CONSTRAINT [DF_BloodTestResults_TakenAfterPropertyRemediationCompleted] DEFAULT ((0)),
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_BloodTestResults_CreatedDate] DEFAULT (getdate()),
[HematocritValue] AS ([hemoglobinValue]* (3)),

```

Project> (local)> User databases> LCCHPDev> Tables> dbo.BloodTestResults

```
[ExcludeResult] [bit] NULL,
[HistoricBloodTestResultsID] [int] NULL,
[HistoricLabResultsID] [varchar] (10) COLLATE SQL_Latin1_General_CI_AS NULL,
[ClientStatusID] [smallint] NULL
) ON [UData]
GO
create trigger [dbo].[trUpdateBloodTestResults] on [dbo].[BloodTestResults] AFTER
UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update BloodTestResults set ModifiedDate = getdate()
where BloodTestResultsID in (select BloodTestResultsID from inserted)

end
GO
ALTER TABLE [dbo].[BloodTestResults] ADD CONSTRAINT [ck_BloodTestResults_SampleDate]
CHECK (((dbo].[udf_DateInThePast]([SampleDate])=(1)))
GO
ALTER TABLE [dbo].[BloodTestResults] ADD CONSTRAINT [PK_BloodTestResults] PRIMARY KEY
CLUSTERED ([BloodTestResultsID]) ON [UData]
GO
CREATE NONCLUSTERED INDEX [IDX_BloodTestResultsSampleDateLeadValue] ON [dbo].[BloodTest-
Results] ([PersonID], [SampleDate], [LeadValue]) INCLUDE ([BloodTestResultsID], [Client-
StatusID], [LabSubmissionDate]) ON [UData]
GO
ALTER TABLE [dbo].[BloodTestResults] ADD CONSTRAINT [FK_BloodTestResults_TargetStatus]
FOREIGN KEY ([ClientStatusID]) REFERENCES [dbo].[TargetStatus] ([StatusID])
GO
ALTER TABLE [dbo].[BloodTestResults] ADD CONSTRAINT [FK_BloodTestResults_Hematocrit-
LevelCategory] FOREIGN KEY ([HematocritValueCategoryID]) REFERENCES [dbo].[SampleLevel-
Category] ([SampleLevelCategoryID])
GO
ALTER TABLE [dbo].[BloodTestResults] ADD CONSTRAINT [FK_BloodTestResults_Hemoglobin-
LevelCategory] FOREIGN KEY ([HemoglobinValueCategoryID]) REFERENCES [dbo].[SampleLevel-
Category] ([SampleLevelCategoryID])
GO
ALTER TABLE [dbo].[BloodTestResults] ADD CONSTRAINT [FK_BloodTestResults_Lab] FOREIGN
KEY ([LabID]) REFERENCES [dbo].[Lab] ([LabID])
GO
ALTER TABLE [dbo].[BloodTestResults] ADD CONSTRAINT [FK_BloodTestResults_LeadLevel-
Category] FOREIGN KEY ([LeadValueCategoryID]) REFERENCES [dbo].[SampleLevelCategory]
([SampleLevelCategoryID])
GO
ALTER TABLE [dbo].[BloodTestResults] ADD CONSTRAINT [FK_BloodTestResults_Person]
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[BloodTestResults] ADD CONSTRAINT [FK_BloodTestResults_SampleType]
FOREIGN KEY ([SampleTypeID]) REFERENCES [dbo].[SampleType] ([SampleTypeID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'Collection of blood test result values
and categorization', 'SCHEMA', N'dbo', 'TABLE', N'BloodTestResults', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'cost of the blood tests', 'SCHEMA',
N'dbo', 'TABLE', N'BloodTestResults', 'COLUMN', N'BloodTestCosts'
GO
EXEC sp_addextendedproperty N'MS_Description', N'uniquie identifier for the blood test
```

Author: liam

```

results object', 'SCHEMA', N'dbo', 'TABLE', N'BloodTestResults', 'COLUMN', N'BloodTest-
ResultsID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the associated hematocrit value
categorization', 'SCHEMA', N'dbo', 'TABLE', N'BloodTestResults', 'COLUMN', N'Hematocrit-
ValueCategoryID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the associated hemoglobin value
categorization', 'SCHEMA', N'dbo', 'TABLE', N'BloodTestResults', 'COLUMN', N'Hemoglobin-
ValueCategoryID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Historic bloodpbresults id from access
database', 'SCHEMA', N'dbo', 'TABLE', N'BloodTestResults', 'COLUMN', N'HistoricBlood-
TestResultsID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'historic lab results id from access
database', 'SCHEMA', N'dbo', 'TABLE', N'BloodTestResults', 'COLUMN', N'HistoricLab-
ResultsID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes', 'SCHEMA', N'dbo',
'TABLE', N'BloodTestResults', 'COLUMN', N'isBaseline'
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the lab to which the samples
were submitted', 'SCHEMA', N'dbo', 'TABLE', N'BloodTestResults', 'COLUMN', N'LabID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the sample was submitted to the
lab', 'SCHEMA', N'dbo', 'TABLE', N'BloodTestResults', 'COLUMN', N'LabSubmissionDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the associated lead value
categorization', 'SCHEMA', N'dbo', 'TABLE', N'BloodTestResults', 'COLUMN', N'LeadValue-
CategoryID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the sample was taken', 'SCHEMA',
N'dbo', 'TABLE', N'BloodTestResults', 'COLUMN', N'SampleDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the type of sample; i.e. venus,
capo, soil, water, nitton analyzer . . . ', 'SCHEMA', N'dbo', 'TABLE', N'BloodTest-
Results', 'COLUMN', N'SampleTypeID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 - No, 1 - yes; was the blood sample
taken after property remediation was completed.', 'SCHEMA', N'dbo', 'TABLE', N'Blood-
TestResults', 'COLUMN', N'TakenAfterPropertyRemediationCompleted'
GO

```

Uses

[dbo].[Lab]
[dbo].[Person]
[dbo].[SampleLevelCategory]
[dbo].[SampleType]
[dbo].[TargetStatus]
[dbo].[udf_DateInThePast]

Used By

[dbo].[BloodTestResultsNotes]

Project> (local)> User databases> LCCHPDev> Tables> dbo.BloodTestResults

```
[dbo].[vMostRecentBloodTestResults]
[dbo].[usp_InsertBloodTestResults]
[dbo].[usp_SLAllBloodTestResultsMetaData]
[dbo].[usp_SICountAdults]
[dbo].[usp_SIISummaryReport]
[dbo].[usp_upBloodTestResults]
[dbo].[usp_upBloodTestResultsWebScreen]
[dbo].[usp_upClientFlag]
```

[dbo].[BloodTestResultsNotes]

MS_Description

linking table for access agreement and access agreement notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	6
Created	12:18:19 AM Tuesday, February 17, 2015
Last Modified	12:18:19 AM Tuesday, February 17, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	BloodTestResultsNotesID	int	4	False	1 - 1	
 D	BloodTestResultsID	int	4	False		
	CreatedDate date the notes where added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	False		

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_BloodTestResultsNotes	BloodTestResults-NotesID	True	UData

Foreign Keys

Name	Columns
FK_BloodTestResultsNotes_BloodTestResults	BloodTestResultsID->[dbo].[BloodTestResults].[BloodTestResultsID]

SQL Script

```
CREATE TABLE [dbo].[BloodTestResultsNotes]
```

Author: liam

```
(  
[BloodTestResultsNotesID] [int] NOT NULL IDENTITY(1, 1),  
[BloodTestResultsID] [int] NOT NULL,  
[CreatedDate] [datetime] NULL CONSTRAINT [DF_BloodTestResultsNotes_CreatedDate] DEFAULT  
(getdate()),  
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NOT NULL  
) ON [UData]  
GO  
ALTER TABLE [dbo].[BloodTestResultsNotes] ADD CONSTRAINT [PK_BloodTestResultsNotes]  
PRIMARY KEY CLUSTERED ([BloodTestResultsNotesID]) ON [UData]  
GO  
ALTER TABLE [dbo].[BloodTestResultsNotes] ADD CONSTRAINT [FK_BloodTestResultsNotes_  
BloodTestResults] FOREIGN KEY ([BloodTestResultsID]) REFERENCES [dbo].[BloodTest-  
Results] ([BloodTestResultsID])  
GO  
EXEC sp_addextendedproperty N'MS_Description', N'linking table for access agreement and  
access agreement notes', 'SCHEMA', N'dbo', 'TABLE', N'BloodTestResultsNotes', NULL,  
NULL  
GO  
EXEC sp_addextendedproperty N'MS_Description', N'date the notes where added', 'SCHEMA',  
N'dbo', 'TABLE', N'BloodTestResultsNotes', 'COLUMN', N'CreatedDate'  
GO
```

Uses

[dbo].[BloodTestResults]

Used By

[dbo].[usp_InsertBloodTestResultsNotes]

[dbo].[CleanupStatus]

MS_Description

collection of clean up status

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	14
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	3:07:34 PM Sunday, April 19, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	CleanupStatusID unique identifier of the cleanup status object	tinyint	1	False	1 - 1	
	CleanupStatusDescription description of the cleanup status	varchar(253)	253	True		
	CleanupStatusName short name for the cleanup status	varchar(50)	50	True		
	HistoricCleanupStatusID	char(1)	1	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_CleanupStatus	CleanupStatusID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateCleanupStatus	True	True	After Update

SQL Script

```

CREATE TABLE [dbo].[CleanupStatus]
(
    [CleanupStatusID] [tinyint] NOT NULL IDENTITY(1, 1),
    [CleanupStatusDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
    [CleanupStatusName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
    [HistoricCleanupStatusID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_CleanupStatus_CreatedDate] DEFAULT
    (getdate())
) ON [UData]
GO
create trigger [dbo].[trUpdateCleanupStatus] on [dbo].[CleanupStatus] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update CleanupStatus set ModifiedDate = getdate()
    where CleanupStatusID in (select CleanupStatusID from inserted)

end
GO
ALTER TABLE [dbo].[CleanupStatus] ADD CONSTRAINT [PK_CleanupStatus] PRIMARY KEY
CLUSTERED ([CleanupStatusID]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of clean up status',
    'SCHEMA', N'dbo', 'TABLE', N'CleanupStatus', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'description of the cleanup status',
    'SCHEMA', N'dbo', 'TABLE', N'CleanupStatus', 'COLUMN', N'CleanupStatusDescription'
GO
EXEC sp_addextendedproperty N'MS_Description', N'unique identifier of the cleanup
status object', 'SCHEMA', N'dbo', 'TABLE', N'CleanupStatus', 'COLUMN', N'CleanupStatus-
ID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'short name for the cleanup status',
    'SCHEMA', N'dbo', 'TABLE', N'CleanupStatus', 'COLUMN', N'CleanupStatusName'
GO

```

Used By

[dbo].[Property]
 [dbo].[PropertytoCleanupStatus]
 [dbo].[usp_InsertCleanupStatus]

[dbo].[Condition]

MS_Description

Collection of potential status for Action

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	0
Created	2:27:18 PM Saturday, April 11, 2015
Last Modified	2:27:18 PM Saturday, April 11, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	ConditionID	tinyint	1	False	1 - 1	
	ConditionDescription Detailed description of the Action status	varchar(253)	253	True		
	ConditionName status for the Action	varchar(50)	50	True		
	HistoricConditionID historic status from access database	char(1)	1	True		
	ModifiedDate last modified date for the record	datetime	8	True		
	CreatedDate date the record was created	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_Condition	ConditionID	True	UData

SQL Script

```
CREATE TABLE [dbo].[Condition]
(
[ConditionID] [tinyint] NOT NULL IDENTITY(1, 1),
```

Author: liam

Project> (local)> User databases> LCCHPDev> Tables> dbo.Condition

```
[ConditionDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[ConditionName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[HistoricConditionID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Condition_CreatedDate] DEFAULT (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[Condition] ADD CONSTRAINT [PK_Condition] PRIMARY KEY CLUSTERED
([ConditionID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Collection of potential status for
Action', 'SCHEMA', N'dbo', 'TABLE', N'Condition', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Detailed description of the Action
status', 'SCHEMA', N'dbo', 'TABLE', N'Condition', 'COLUMN', N'ConditionDescription'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'status for the Action', 'SCHEMA',
N'dbo', 'TABLE', N'Condition', 'COLUMN', N'ConditionName'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the record was created',
'SCHEMA', N'dbo', 'TABLE', N'Condition', 'COLUMN', N'CreatedDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'historic status from access database',
'SCHEMA', N'dbo', 'TABLE', N'Condition', 'COLUMN', N'HistoricConditionID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'last modified date for the record',
'SCHEMA', N'dbo', 'TABLE', N'Condition', 'COLUMN', N'ModifiedDate'
GO
```

[dbo].[ConstructionType]

MS_Description

collection of construction types

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	10
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	4:55:06 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	ConstructionTypeID unique identifier of the construction type	tinyint	1	False	1 - 1	
	ConstructionTypeName description of the construction type	varchar(50)	50	False		
	ConstructionTypeDescription	varchar(253)	253	True		
	HistoricConstructionTypeID	char(1)	1	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_ConstructionType	ConstructionTypeID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateConstructionType	True	True	After Update

SQL Script

```

CREATE TABLE [dbo].[ConstructionType]
(
    [ConstructionTypeID] [tinyint] NOT NULL IDENTITY(1, 1),
    [ConstructionTypeName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
    [ConstructionTypeDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
    [HistoricConstructionTypeID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_ConstructionType_CreatedDate] DEFAULT
        (getdate())
    ) ON [UData]
GO
create trigger [dbo].[trUpdateConstructionType] on [dbo].[ConstructionType] AFTER
UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update ConstructionType set ModifiedDate = getdate()
    where ConstructionTypeID in (select ConstructionTypeID from inserted)

end
GO
ALTER TABLE [dbo].[ConstructionType] ADD CONSTRAINT [PK_ConstructionType] PRIMARY KEY
CLUSTERED ([ConstructionTypeID]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of construction types',
    'SCHEMA', N'dbo', 'TABLE', N'ConstructionType', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'unique identifier of the construction
    type', 'SCHEMA', N'dbo', 'TABLE', N'ConstructionType', 'COLUMN', N'ConstructionTypeID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'description of the construction type',
    'SCHEMA', N'dbo', 'TABLE', N'ConstructionType', 'COLUMN', N'ConstructionTypeName'
GO

```

Used By

[dbo].[Property]
 [dbo].[usp_InsertConstructionType]

[dbo].[ContactType]

MS_Description

Collection of contact types

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	0
Created	6:17:34 PM Thursday, April 16, 2015
Last Modified	6:17:34 PM Thursday, April 16, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	ContactTypeID	tinyint	1	False	1 - 1	
	ContactTypeDescription Detailed description of the Action status	varchar(253)	253	True		
	ContactTypeName status for the Action	varchar(50)	50	True		
	HistoricContactTypeID historic status from access database	char(1)	1	True		
	ModifiedDate last modified date for the record	datetime	8	True		
	CreatedDate date the record was created	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_ContactType	ContactTypeID	True	UData

SQL Script

```
CREATE TABLE [dbo].[ContactType]
(
[ContactTypeID] [tinyint] NOT NULL IDENTITY(1, 1),
```

Author: liam

Project> (local)> User databases> LCCHPDev> Tables> dbo.ContactType

```
[ContactTypeID] [int] IDENTITY(1,1) NOT NULL,
[ContactTypeDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[ContactTypeName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[HistoricContactTypeID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_ContactType_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[ContactType] ADD CONSTRAINT [PK_ContactType] PRIMARY KEY CLUSTERED
([ContactTypeID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Collection of contact types',
'SCHEMA', N'dbo', 'TABLE', N'ContactType', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Detailed description of the Action
status', 'SCHEMA', N'dbo', 'TABLE', N'ContactType', 'COLUMN', N>ContactTypeDescription'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'status for the Action', 'SCHEMA',
N'dbo', 'TABLE', N'ContactType', 'COLUMN', N>ContactTypeName'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the record was created',
'SCHEMA', N'dbo', 'TABLE', N'ContactType', 'COLUMN', N'CreatedDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'historic status from access database',
'SCHEMA', N'dbo', 'TABLE', N'ContactType', 'COLUMN', N'HistoricContactTypeID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'last modified date for the record',
'SCHEMA', N'dbo', 'TABLE', N'ContactType', 'COLUMN', N'ModifiedDate'
GO
```

[dbo].[Contractor]	

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	2
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	4:55:06 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	ContractorID	int	4	False	1 - 1	
	ContractorName	varchar(50)	50	True		
	ContractorDescription	varchar(253)	253	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_Contractor	ContractorID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateContractor	True	True	After Update

SQL Script

```
CREATE TABLE [dbo].[Contractor]
(
[ContractorID] [int] NOT NULL IDENTITY(1, 1),
[ContractorName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[ContractorDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Contractor_CreatedDate] DEFAULT
(getdate())
)
```

Author: liam

```
) ON [UData]
GO
create trigger [dbo].[trUpdateContractor] on [dbo].[Contractor] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update Contractor set ModifiedDate = getdate() where
ContractorID in (select ContractorID from inserted)

    end
GO
ALTER TABLE [dbo].[Contractor] ADD CONSTRAINT [PK_Contractor] PRIMARY KEY CLUSTERED
([ContractorID]) ON [UData]
GO
```

Used By

[dbo].[ContractortoProperty]
[dbo].[ContractortoRemediation]
[dbo].[ContractortoRemediationActionPlan]
[dbo].[usp_InsertContractor]

[dbo].[ContractortoProperty]	

MS_Description

linking table for contractor and occupied properties

Properties

Property	Value
File Group	UData
Row Count (~)	2
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	2:04:12 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	ContractorID	int	4	False	
	PropertyID	int	4	False	
	StartDate date the contractor started occupying the property	date	3	True	
	EndDate date contractor ended property occupation	date	3	True	
	ModifiedDate	datetime	8	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_ContractortoProperty	ContractorID, Property-ID	True	UData

Foreign Keys

Name	Columns
FK_ContractortoProperty_Contractor building which the contractor occupies for purpose of business (contractor offices)	ContractorID->[dbo].[Contractor].[ContractorID]

FK_ContractortoProperty_Property	PropertyID->[dbo].[Property].[PropertyID]
----------------------------------	---

SQL Script

```

CREATE TABLE [dbo].[ContractortoProperty]
(
    [ContractorID] [int] NOT NULL,
    [PropertyID] [int] NOT NULL,
    [StartDate] [date] NULL,
    [EndDate] [date] NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_ContractortoProperty_CreatedDate] DEFAULT
    (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[ContractortoProperty] ADD CONSTRAINT [PK_ContractortoProperty]
PRIMARY KEY CLUSTERED ([ContractorID], [PropertyID]) ON [UData]
GO
ALTER TABLE [dbo].[ContractortoProperty] ADD CONSTRAINT [FK_ContractortoProperty_-
Contractor] FOREIGN KEY ([ContractorID]) REFERENCES [dbo].[Contractor] ([ContractorID])
GO
ALTER TABLE [dbo].[ContractortoProperty] ADD CONSTRAINT [FK_ContractortoProperty_-
Property] FOREIGN KEY ([PropertyID]) REFERENCES [dbo].[Property] ([PropertyID])
GO
EXEC sp_adddextendedproperty N'MS_Description', N'linking table for contractor and
occupied properties', 'SCHEMA', N'dbo', 'TABLE', N'ContractortoProperty', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date contractor ended property
occupation', 'SCHEMA', N'dbo', 'TABLE', N'ContractortoProperty', 'COLUMN', N'EndDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the contractor started occupying
the property', 'SCHEMA', N'dbo', 'TABLE', N'ContractortoProperty', 'COLUMN', N'Start-
Date'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'building which the contractor occupies
for purpose of business (contractor offices)', 'SCHEMA', N'dbo', 'TABLE',
N'ContractortoProperty', 'CONSTRAINT', N'FK_ContractortoProperty_Contractor'
GO

```

Uses

[dbo].[Contractor]
 [dbo].[Property]

Used By

[dbo].[usp_InsertContractortoProperty]

[dbo].[ContractortoRemediation]	

MS_Description

linking table for contractors and remediations

Properties

Property	Value
File Group	UData
Row Count (~)	1
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	2:08:33 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	ContractorID	int	4	False	
	RemediationID	int	4	False	
	StartDate date the contractor started working on the remediation	date	3	True	
	EndDate date the contractor stopped working on the remediation	date	3	True	
	isSubContractor 0 - no, 1 - yes. is this contractor a sub contractor	bit	1	True	
	ModifiedDate	datetime	8	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_ContractortoRemediation	ContractorID, RemediationID	True	UData

Foreign Keys

Name	Columns
FK_ContractortoRemediation_Contractor	ContractorID->[dbo].[Contractor].[ContractorID]
FK_ContractortoRemediation_Remediation	RemediationID->[dbo].[Remediation].[RemediationID]

SQL Script

```

CREATE TABLE [dbo].[ContractortoRemediation]
(
    [ContractorID] [int] NOT NULL,
    [RemediationID] [int] NOT NULL,
    [StartDate] [date] NULL,
    [EndDate] [date] NULL,
    [isSubContractor] [bit] NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_ContractortoRemediation_CreatedDate]
        DEFAULT (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[ContractortoRemediation] ADD CONSTRAINT [PK_ContractortoRemediation]
PRIMARY KEY CLUSTERED ([ContractorID], [RemediationID]) ON [UData]
GO
ALTER TABLE [dbo].[ContractortoRemediation] ADD CONSTRAINT [FK_ContractortoRemediation_Contractor]
FOREIGN KEY ([ContractorID]) REFERENCES [dbo].[Contractor] ([ContractorID])
GO
ALTER TABLE [dbo].[ContractortoRemediation] ADD CONSTRAINT [FK_ContractortoRemediation_Remediation]
FOREIGN KEY ([RemediationID]) REFERENCES [dbo].[Remediation] ([RemediationID])
GO
EXEC sp_adddextendedproperty N'MS_Description', N'linking table for contractors and
remediations', 'SCHEMA', N'dbo', 'TABLE', N'ContractortoRemediation', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the contractor stopped working on
the remediation', 'SCHEMA', N'dbo', 'TABLE', N'ContractortoRemediation', 'COLUMN',
N'EndDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'0 - no, 1 - yes. is this contractor a
sub contractor', 'SCHEMA', N'dbo', 'TABLE', N'ContractortoRemediation', 'COLUMN', N'is-
SubContractor'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the contractor started working on
the remediation', 'SCHEMA', N'dbo', 'TABLE', N'ContractortoRemediation', 'COLUMN',
N'StartDate'
GO

```

Uses

[dbo].[Contractor]
 [dbo].[Remediation]

Used By

[dbo].[usp_InsertContractortoRemediation]

[dbo].[ContractortoRemediationActionPlan]

MS_Description

linking table for contractor and sampling plan

Properties

Property	Value
File Group	UData
Row Count (~)	1
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	2:08:33 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
PK	ContractorID	int	4	False	
PK	RemediationActionPlanID	int	4	False	
	StartDate	date	3	True	
	EndDate	date	3	True	
	isSubContractor	bit	1	True	
	ModifiedDate	datetime	8	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
PK	PK_ContractortoRemediationActionPlan	ContractorID, RemediationAction-PlanID	True	UData

Foreign Keys

Name	Columns
FK_ContractortoRemediationActionPlan_Contractor	ContractorID->[dbo].[Contractor].[ContractorID]
FK_ContractortoSamplingPlan_Contractor	ContractorID->[dbo].[Contractor].[ContractorID]
FK_ContractortoRemediationActionPlan_Remediation-ActionPlan	RemediationActionPlanID->[dbo].[RemediationAction-Plan].[RemediationActionPlanID]

FK_ContractortoRemediationPlan_RemediationAction-Plan	RemediationActionPlanID->[dbo].[RemediationAction-Plan].[RemediationActionPlanID]
---	---

SQL Script

```

CREATE TABLE [dbo].[ContractortoRemediationActionPlan]
(
    [ContractorID] [int] NOT NULL,
    [RemediationActionPlanID] [int] NOT NULL,
    [StartDate] [date] NULL,
    [EndDate] [date] NULL,
    [isSubContractor] [bit] NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_ContractortoRemediationActionPlan_Created-Date] DEFAULT (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[ContractortoRemediationActionPlan] ADD CONSTRAINT [PK_Contractorto-RemediationActionPlan] PRIMARY KEY CLUSTERED ([ContractorID], [RemediationActionPlan-ID]) ON [UData]
GO
ALTER TABLE [dbo].[ContractortoRemediationActionPlan] ADD CONSTRAINT [FK_Contractorto-RemediationActionPlan_Contractor] FOREIGN KEY ([ContractorID]) REFERENCES [dbo].[Contractor] ([ContractorID])
GO
ALTER TABLE [dbo].[ContractortoRemediationActionPlan] ADD CONSTRAINT [FK_Contractorto-SamplingPlan_Contractor] FOREIGN KEY ([ContractorID]) REFERENCES [dbo].[Contractor] ([ContractorID])
GO
ALTER TABLE [dbo].[ContractortoRemediationActionPlan] ADD CONSTRAINT [FK_Contractorto-RemediationActionPlan_RemediationActionPlan] FOREIGN KEY ([RemediationActionPlanID]) REFERENCES [dbo].[RemediationActionPlan] ([RemediationActionPlanID])
GO
ALTER TABLE [dbo].[ContractortoRemediationActionPlan] ADD CONSTRAINT [FK_Contractorto-RemediationActionPlan_RemediationActionPlan] FOREIGN KEY ([RemediationActionPlanID]) REFERENCES [dbo].[RemediationActionPlan] ([RemediationActionPlanID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for contractor and sampling plan', 'SCHEMA', 'dbo', 'TABLE', 'N'ContractortoRemediationActionPlan', NULL, NULL
GO

```

Uses

[dbo].[Contractor]
 [dbo].[RemediationActionPlan]

Used By

[dbo].[usp_InsertContractortoRemediationActionPlan]

[dbo].[Country]

MS_Description

collection of countries

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	3
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	4:55:06 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	CountryID unique identifier of the country	tinyint	1	False	1 - 1	
	CountryName name of the country	varchar(50)	50	False		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_Country	CountryID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateCountry	True	True	After Update

SQL Script

```
CREATE TABLE [dbo].[Country]
(

```

Author: liam

```
[CountryID] [tinyint] NOT NULL IDENTITY(1, 1),
[CountryName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Country_CreatedDate] DEFAULT (getdate())
) ON [UData]
GO
create trigger [dbo].[trUpdateCountry] on [dbo].[Country] AFTER UPDATE
as
begin
if @@rowcount = 0
return
if not update(ModifiedDate) update Country set ModifiedDate = getdate() where
CountryID in (select CountryID from inserted)

end
GO
ALTER TABLE [dbo].[Country] ADD CONSTRAINT [PK_Country] PRIMARY KEY CLUSTERED
([CountryID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'collection of countries', 'SCHEMA',
N'dbo', 'TABLE', N'Country', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'unique identifier of the country',
'SCHEMA', N'dbo', 'TABLE', N'Country', 'COLUMN', N'CountryID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'name of the country', 'SCHEMA',
N'dbo', 'TABLE', N'Country', 'COLUMN', N'CountryName'
GO
```

Used By

[dbo].[ForeignFoodtoCountry]
[dbo].[PersonToTravelCountry]
[dbo].[usp_InsertCountry]

[dbo].[DataSource]

MS_Description

Collection of contact types

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	0
Created	2:40:10 PM Saturday, April 11, 2015
Last Modified	2:40:10 PM Saturday, April 11, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	DataSourceID	tinyint	1	False	1 - 1	
	DataSourceDescription Detailed description of the data source	varchar(253)	253	True		
	DataSourceName short name for the data source	varchar(50)	50	True		
	HistoricDataSourceID historic data source ID from access database	char(1)	1	True		
	ModifiedDate last modified date for the record	datetime	8	True		
	CreatedDate date the record was created	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_DataSource	DataSourceID	True	UData

SQL Script

```
CREATE TABLE [dbo].[DataSource]
(
```

Author: liam

Project> (local)> User databases> LCCHPDev> Tables> dbo.DataSource

```
[DataSourceID] [tinyint] NOT NULL IDENTITY(1, 1),
[DataSourceDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[DataSourceName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[HistoricDataSourceID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_DataSource_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[DataSource] ADD CONSTRAINT [PK_DataSource] PRIMARY KEY CLUSTERED
([DataSourceID]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Collection of contact types',
'SCHEMA', N'dbo', 'TABLE', N'DataSource', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the record was created',
'SCHEMA', N'dbo', 'TABLE', N'DataSource', 'COLUMN', N'CreatedDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Detailed description of the data
source', 'SCHEMA', N'dbo', 'TABLE', N'DataSource', 'COLUMN', N'DataSourceDescription'
GO
EXEC sp_addextendedproperty N'MS_Description', N'short name for the data source',
'SCHEMA', N'dbo', 'TABLE', N'DataSource', 'COLUMN', N'DataSourceName'
GO
EXEC sp_addextendedproperty N'MS_Description', N'historic data source ID from access
database', 'SCHEMA', N'dbo', 'TABLE', N'DataSource', 'COLUMN', N'HistoricDataSourceID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'last modified date for the record',
'SCHEMA', N'dbo', 'TABLE', N'DataSource', 'COLUMN', N'ModifiedDate'
GO
```

[dbo].[Daycare]

MS_Description

collection of daycare facilities

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	6
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	11:53:03 PM Monday, April 27, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	DaycareID	int	4	False	1 - 1	
	DaycareName name of the daycare	varchar(50)	50	False		
	DaycareDescription short description of the daycare business	varchar(253)	253	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_Daycare	DaycareID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateDaycare	True	True	After Update

SQL Script

```

CREATE TABLE [dbo].[Daycare]
(
[DaycareID] [int] NOT NULL IDENTITY(1, 1),
[DaycareName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
[DaycareDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Daycare_CreatedDate] DEFAULT (getdate())
) ON [UData]
GO
create trigger [dbo].[trUpdateDaycare] on [dbo].[Daycare] AFTER UPDATE
as
begin
if @@rowcount = 0
return
if not update(ModifiedDate) update Daycare set ModifiedDate = getdate() where
DaycareID in (select DaycareID from inserted)

end
GO
ALTER TABLE [dbo].[Daycare] ADD CONSTRAINT [PK_Daycare] PRIMARY KEY CLUSTERED
([DaycareID]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of daycare facilities',
'SCHEMA', N'dbo', 'TABLE', N'Daycare', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'short description of the daycare
business', 'SCHEMA', N'dbo', 'TABLE', N'Daycare', 'COLUMN', N'DaycareDescription'
GO
EXEC sp_addextendedproperty N'MS_Description', N'name of the daycare', 'SCHEMA',
N'dbo', 'TABLE', N'Daycare', 'COLUMN', N'DaycareName'
GO

```

Used By

- [dbo].[DaycaretoProperty]
- [dbo].[PersontoDaycare]
- [dbo].[Questionnaire]
- [dbo].[usp_InsertDaycare]
- [dbo].[usp_SIDaycare]

[dbo].[DaycarePrimaryContact]

MS_Description

linking table for daycare and person - identifying contact person

Properties

Property	Value
File Group	UData
Row Count (~)	3
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	6:58:11 PM Saturday, April 4, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	DaycareID	int	4	False	
	PersonID	int	4	False	
	ContactPriority priority of this person in the contact list (1 being highest priority)	tinyint	1	False	((1))
	PrimaryPhoneNumberID id of the primary contact number	int	4	True	
	ModifiedDate	datetime	8	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_DaycareContactPerson	DaycareID, PersonID, ContactPriority	True	UData

SQL Script

```
CREATE TABLE [dbo].[DaycarePrimaryContact]
(
[DaycareID] [int] NOT NULL,
[PersonID] [int] NOT NULL,
[ContactPriority] [tinyint] NOT NULL CONSTRAINT [DF_DaycareContactPerson_Contact-]
```

Author: liam

```
Priority] DEFAULT ((1)),
[PrimaryPhoneNumberID] [int] NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_DaycarePrimaryContact_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[DaycarePrimaryContact] ADD CONSTRAINT [PK_DaycareContactPerson]
PRIMARY KEY CLUSTERED ([DaycareID], [PersonID], [ContactPriority]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for daycare and person -
identifying contact person', 'SCHEMA', N'dbo', 'TABLE', N'DaycarePrimaryContact', NULL,
NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'priority of this person in the contact
list (1 being highest priority)', 'SCHEMA', N'dbo', 'TABLE', N'DaycarePrimaryContact',
'COLUMN', N>ContactPriority'
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the primary contact number',
'SCHEMA', N'dbo', 'TABLE', N'DaycarePrimaryContact', 'COLUMN', N'PrimaryPhoneNumberID'
GO
```

Used By

[dbo].[usp_InsertDaycarePrimaryContact]

[dbo].[DaycaretoProperty]

MS_Description

linking table for daycare and property

Properties

Property	Value
File Group	UData
Row Count (~)	2
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	2:08:33 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	DaycareID	int	4	False	
	PropertyID	int	4	False	
	StartDate date the daycare started occupying the property	date	3	False	(getdate())
	EndDate date the daycare stopped occupying the property	date	3	True	
	ModifiedDate	datetime	8	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_DaycaretoProperty	DaycareID, PropertyID, StartDate	True	UData

Foreign Keys

Name	Columns
FK_DaycaretoProperty_Daycare	DaycareID->[dbo].[Daycare].[DaycareID]

FK_DaycaretoProperty_Property	PropertyID->[dbo].[Property].[PropertyID]
-------------------------------	---

SQL Script

```

CREATE TABLE [dbo].[DaycaretoProperty]
(
    [DaycareID] [int] NOT NULL,
    [PropertyID] [int] NOT NULL,
    [StartDate] [date] NOT NULL CONSTRAINT [DF_DaycaretoProperty_StartDate] DEFAULT
    (getdate()),
    [EndDate] [date] NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_DaycaretoProperty_CreatedDate] DEFAULT
    (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[DaycaretoProperty] ADD CONSTRAINT [PK_DaycaretoProperty] PRIMARY KEY
CLUSTERED ([DaycareID], [PropertyID], [StartDate]) ON [UData]
GO
ALTER TABLE [dbo].[DaycaretoProperty] ADD CONSTRAINT [FK_DaycaretoProperty_Daycare]
FOREIGN KEY ([DaycareID]) REFERENCES [dbo].[Daycare] ([DaycareID])
GO
ALTER TABLE [dbo].[DaycaretoProperty] ADD CONSTRAINT [FK_DaycaretoProperty_Property]
FOREIGN KEY ([PropertyID]) REFERENCES [dbo].[Property] ([PropertyID])
GO
EXEC sp_adddextendedproperty N'MS_Description', N'linking table for daycare and
property', 'SCHEMA', N'dbo', 'TABLE', N'DaycaretoProperty', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the daycare stopped occupying the
property', 'SCHEMA', N'dbo', 'TABLE', N'DaycaretoProperty', 'COLUMN', N'EndDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the daycare started occupying the
property', 'SCHEMA', N'dbo', 'TABLE', N'DaycaretoProperty', 'COLUMN', N'StartDate'
GO

```

Uses

[dbo].[Daycare]
[dbo].[Property]

Used By

[dbo].[usp_InsertDaycaretoProperty]

[dbo].[Employer]

MS_Description

collection of employers

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	5
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	4:55:06 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	EmployerID unique identifier of the employer	int	4	False	1 - 1	
	EmployerName name of the employer	varchar(50)	50	False		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_Employer	EmployerID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateEmployer	True	True	After Update

SQL Script

```
CREATE TABLE [dbo].[Employer]
(

```

Author: liam

```
[EmployerID] [int] NOT NULL IDENTITY(1, 1),
[EmployerName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
[ModifiedDate] [datetime] NULL CONSTRAINT [DF_Employer_CreatedDate] DEFAULT (getdate())
) ON [UData]
GO
create trigger [dbo].[trUpdateEmployer] on [dbo].[Employer] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update Employer set ModifiedDate = getdate() where
EmployerID in (select EmployerID from inserted)

    end
GO
ALTER TABLE [dbo].[Employer] ADD CONSTRAINT [PK_Employer] PRIMARY KEY CLUSTERED
([EmployerID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'collection of employers', 'SCHEMA',
N'dbo', 'TABLE', N'Employer', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'unique identifier of the employer',
'SCHEMA', N'dbo', 'TABLE', N'Employer', 'COLUMN', N'EmployerID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'name of the employer', 'SCHEMA',
N'dbo', 'TABLE', N'Employer', 'COLUMN', N'EmployerName'
GO
```

Used By

[dbo].[EmployertoProperty]
[dbo].[PersonstoEmployer]
[dbo].[usp_InsertEmployer]

[dbo].[EmployertoProperty]	

MS_Description

linking table for employer and property

Properties

Property	Value
File Group	UData
Row Count (~)	2
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	2:08:33 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	EmployerID	int	4	False	
	PropertyID	int	4	False	
	StartDate date the employer started occupying the property	date	3	False	(getdate())
	EndDate date the employer stopped occupying the property	date	3	True	
	ModifiedDate	datetime	8	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_EmployertoProperty	EmployerID, PropertyID, StartDate	True	UData

Foreign Keys

Name	Columns
FK_EmployertoProperty_Employer	EmployerID->[dbo].[Employer].[EmployerID]

FK_EmployertoProperty_Property	PropertyID->[dbo].[Property].[PropertyID]
--------------------------------	---

SQL Script

```

CREATE TABLE [dbo].[EmployertoProperty]
(
    [EmployerID] [int] NOT NULL,
    [PropertyID] [int] NOT NULL,
    [StartDate] [date] NOT NULL CONSTRAINT [DF_EmployertoProperty_StartDate] DEFAULT
    (getdate()),
    [EndDate] [date] NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_EmployertoProperty_CreatedDate] DEFAULT
    (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[EmployertoProperty] ADD CONSTRAINT [PK_EmployertoProperty] PRIMARY
KEY CLUSTERED ([EmployerID], [PropertyID], [StartDate]) ON [UData]
GO
ALTER TABLE [dbo].[EmployertoProperty] ADD CONSTRAINT [FK_EmployertoProperty_Employer]
FOREIGN KEY ([EmployerID]) REFERENCES [dbo].[Employer] ([EmployerID])
GO
ALTER TABLE [dbo].[EmployertoProperty] ADD CONSTRAINT [FK_EmployertoProperty_Property]
FOREIGN KEY ([PropertyID]) REFERENCES [dbo].[Property] ([PropertyID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for employer and
property', 'SCHEMA', N'dbo', 'TABLE', N'EmployertoProperty', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the employer stopped occupying
the property', 'SCHEMA', N'dbo', 'TABLE', N'EmployertoProperty', 'COLUMN', N'EndDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the employer started occupying
the property', 'SCHEMA', N'dbo', 'TABLE', N'EmployertoProperty', 'COLUMN', N'StartDate'
GO

```

Uses

[dbo].[Employer]
[dbo].[Property]

Used By

[dbo].[usp_InsertEmployertoProperty]

[dbo].[EnvironmentalInvestigation]	

Properties

Property	Value
File Group	UData
Row Count (~)	4
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	4:55:07 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	EnvironmentalInvestigationID	int	4	False	1 - 1	
	ConductEnvironmentalInvestigation 0 - no, 1 - yes; is an environmental investigation going to be conducted	bit	1	True		
	ConductEnvironmentalInvestigation-DecisionDate date the workgroup decided whether to conduct an environmental investigation or not	date	3	True		
	Cost cost of the environmental investigation	money	8	True		
	EnvironmentalInvestigationDate	date	3	True		
 D	PropertyID	int	4	False		
	StartDate	date	3	True		
	EndDate	date	3	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
 E	PK_EnvironmentalInvestigation	EnvironmentalInvestigationID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateEnvironmentalInvestigation	True	True	After Update

Foreign Keys

Name	Columns
FK_EnvironmentalInvestigation_Property	PropertyID->[dbo].[Property].[PropertyID]

SQL Script

```

CREATE TABLE [dbo].[EnvironmentalInvestigation]
(
[EnvironmentalInvestigationID] [int] NOT NULL IDENTITY(1, 1),
[ConductEnvironmentalInvestigation] [bit] NULL,
[ConductEnvironmentalInvestigationDecisionDate] [date] NULL,
[Cost] [money] NULL,
[EnvironmentalInvestigationDate] [date] NULL,
[PropertyID] [int] NOT NULL,
[StartDate] [date] NULL,
[EndDate] [date] NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_EnvironmentalInvestigation_CreatedDate]
DEFAULT (getdate())
) ON [UData]
GO
create trigger [dbo].[trUpdateEnvironmentalInvestigation] on [dbo].[Environmental-
Investigation] AFTER UPDATE
as
begin
if @@rowcount = 0
    return
    if not update(ModifiedDate) update EnvironmentalInvestigation set ModifiedDate =
getdate() where EnvironmentalInvestigationID in (select EnvironmentalInvestigationID
from inserted)

end
GO
ALTER TABLE [dbo].[EnvironmentalInvestigation] ADD CONSTRAINT [PK_Environmental-
Investigation] PRIMARY KEY CLUSTERED ([EnvironmentalInvestigationID]) ON [UData]
GO
ALTER TABLE [dbo].[EnvironmentalInvestigation] ADD CONSTRAINT [FK_Environmental-
Investigation_Property] FOREIGN KEY ([PropertyID]) REFERENCES [dbo].[Property]
([PropertyID])
GO
EXEC sp_adddextendedproperty N'MS_Description', N'0 - no, 1 - yes; is an environmental
investigation going to be conducted', 'SCHEMA', N'dbo', 'TABLE', N'Environmental-
Investigation', 'COLUMN', N'ConductEnvironmentalInvestigation'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the workgroup decided whether to
conduct an environmental investigation or not', 'SCHEMA', N'dbo', 'TABLE',
N'EnvironmentalInvestigation', 'COLUMN', N'ConductEnvironmentalInvestigationDecision-

```

```
Date'  
GO  
EXEC sp_addextendedproperty N'MS_Description', N'cost of the environmental  
investigation', 'SCHEMA', N'dbo', 'TABLE', N'EnvironmentalInvestigation', 'COLUMN',  
N'Cost'  
GO
```

Uses

[dbo].[Property]

Used By

[dbo].[RemediationActionPlan]

[dbo].[usp_InsertEnvironmentalInvestigation]

[dbo].[ErrorLog]	

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	815
Created	10:31:53 PM Tuesday, December 23, 2014
Last Modified	2:08:33 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	ErrorID	int	4	False	1 - 1	
	Username	nvarchar(128)	256	False		
	ErrorNumber	int	4	True		
	ErrorSeverity	int	4	True		
	ErrorState	int	4	True		
	ErrorProcedure	nvarchar(128)	256	True		
	ErrorLine	int	4	True		
	ErrorMessage	nvarchar(4000)	8000	True		
	ErrorTime	datetime	8	True		(getdate())
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_Error	ErrorID	True	UData

SQL Script

```
CREATE TABLE [dbo].[ErrorLog]
(
[ErrorID] [int] NOT NULL IDENTITY(1, 1),
[Username] [nvarchar] (128) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
[ErrorNumber] [int] NULL,
[ErrorSeverity] [int] NULL,
```

```
[ErrorState] [int] NULL,  
[ErrorProcedure] [nvarchar] (128) COLLATE SQL_Latin1_General_CI_AS NULL,  
[ErrorLine] [int] NULL,  
[ErrorMessage] [nvarchar] (4000) COLLATE SQL_Latin1_General_CI_AS NULL,  
[ErrorTime] [datetime] NULL CONSTRAINT [DF_ErrorLog_ErrorTime] DEFAULT (getdate()),  
[ModifiedDate] [datetime] NULL,  
[CreatedDate] [datetime] NULL CONSTRAINT [DF_ErrorLog_CreatedDate] DEFAULT (getdate())  
) ON [UData]  
GO  
ALTER TABLE [dbo].[ErrorLog] ADD CONSTRAINT [PK_Error] PRIMARY KEY CLUSTERED ([Error-  
ID]) ON [UData]  
GO
```

Used By

[dbo].[usp.LogError]

[dbo].[Ethnicity]

MS_Description

collection of ethnicities

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	8
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	4:55:07 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
PK_C	EthnicityID unique identifier of ethnicities	tinyint	1	False	1 - 1	
	Ethnicity friendly shortname of ethnicity	varchar(50)	50	False		
	HistoricEthnicityCode	char(1)	1	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
PK_C	PK_Ethnicity	EthnicityID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateEthnicity	True	True	After Update

SQL Script

```

CREATE TABLE [dbo].[Ethnicity]
(
    [EthnicityID] [tinyint] NOT NULL IDENTITY(1, 1),
    [Ethnicity] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
    [HistoricEthnicityCode] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_Ethnicity_CreatedDate] DEFAULT (getdate())
) ON [UData]
GO
create trigger [dbo].[trUpdateEthnicity] on [dbo].[Ethnicity] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update Ethnicity set ModifiedDate = getdate() where
EthnicityID in (select EthnicityID from inserted)

    end
GO
ALTER TABLE [dbo].[Ethnicity] ADD CONSTRAINT [PK_Ethnicity] PRIMARY KEY CLUSTERED
([EthnicityID]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of ethnicities', 'SCHEMA',
N'dbo', 'TABLE', N'Ethnicity', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'friendly shortname of ethnicity',
'SCHEMA', N'dbo', 'TABLE', N'Ethnicity', 'COLUMN', N'Ethnicity'
GO
EXEC sp_addextendedproperty N'MS_Description', N'unique identifier of ethnicities',
'SCHEMA', N'dbo', 'TABLE', N'Ethnicity', 'COLUMN', N'EthnicityID'
GO

```

Used By

[dbo].[PersontoEthnicity]
[dbo].[usp_InsertEthnicity]

[dbo].[Family]

MS_Description

collection of families

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	2028
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	9:53:25 AM Saturday, April 18, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	FamilyID unique identifier for the family object	int	4	False	1 - 1	
	Lastname family name	varchar(50)	50	False		
	NumberofSmokers number of smokers in the family	tinyint	1	True		
	PrimaryLanguageID id of the families primary language; default = 1 (English)	tinyint	1	True		((1))
	Pets	tinyint	1	True		
	Petsinandout	bit	1	True		
	HistoricFamilyID	smallint	2	True		
	PrimaryPropertyID	int	4	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())
	FrequentlyWashPets	bit	1	True		
	ForeignTravel does the family travel to foreign countries	bit	1	True		
	ReviewStatusID Review status id	tinyint	1	True		

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_Family	FamilyID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateFamily	True	True	After Update

SQL Script

```

CREATE TABLE [dbo].[Family]
(
    [FamilyID] [int] NOT NULL IDENTITY(1, 1),
    [Lastname] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
    [NumberofSmokers] [tinyint] NULL,
    [PrimaryLanguageID] [tinyint] NULL CONSTRAINT [DF_Family_PrimaryLanguageID] DEFAULT
    ((1)),
    [Pets] [tinyint] NULL,
    [Petsonandout] [bit] NULL,
    [HistoricFamilyID] [smallint] NULL,
    [PrimaryPropertyID] [int] NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_Family_CreatedDate] DEFAULT (getdate()),
    [FrequentlyWashPets] [bit] NULL,
    [ForeignTravel] [bit] NULL,
    [ReviewStatusID] [tinyint] NULL
) ON [UData]
GO
create trigger [dbo].[trUpdateFamily] on [dbo].[Family] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update Family set ModifiedDate = getdate() where
FamilyID in (select FamilyID from inserted)

    end
GO
ALTER TABLE [dbo].[Family] ADD CONSTRAINT [PK_Family] PRIMARY KEY CLUSTERED ([Family-
ID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'collection of families', 'SCHEMA',
N'dbo', 'TABLE', N'Family', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'unique identifier for the family
object', 'SCHEMA', N'dbo', 'TABLE', N'Family', 'COLUMN', N'FamilyID'
GO

```

```
EXEC sp_addextendedproperty N'MS_Description', N'does the family travel to foreign
countries', 'SCHEMA', N'dbo', 'TABLE', N'Family', 'COLUMN', N'ForeignTravel'
GO
EXEC sp_addextendedproperty N'MS_Description', N'family name', 'SCHEMA', N'dbo',
'TABLE', N'Family', 'COLUMN', N'Lastname'
GO
EXEC sp_addextendedproperty N'MS_Description', N'number of smokers in the family',
'SCHEMA', N'dbo', 'TABLE', N'Family', 'COLUMN', N'NumberofSmokers'
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the families primary language;
default = 1 (English)', 'SCHEMA', N'dbo', 'TABLE', N'Family', 'COLUMN', N'Primary-
LanguageID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Review status id', 'SCHEMA', N'dbo',
'TABLE', N'Family', 'COLUMN', N'ReviewStatusID'
GO
```

Used By

- [dbo].[FamilyNotes]
- [dbo].[FamilytoPhoneNumber]
- [dbo].[FamilytoProperty]
- [dbo].[PersontoFamily]
- [dbo].[TravelNotes]
- [dbo].[usp_InsertFamily]
- [dbo].[usp_InsertNewClientWebScreen]
- [dbo].[usp_InsertNewFamilyWebScreen]
- [dbo].[usp_SIInsertedDataSimplified]
- [dbo].[usp_upClientWebScreen]
- [dbo].[usp_upFamily]
- [dbo].[usp_upFamilyWebScreen]
- [dbo].[udf_SIFamilyPhoneNumber]

[dbo].[FamilyNotes]

MS_Description

table for Family notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	127
Created	11:10:54 AM Saturday, February 14, 2015
Last Modified	11:10:54 AM Saturday, February 14, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	FamilyNotesID	int	4	False	1 - 1	
	FamilyID	int	4	False		
	CreatedDate date the notes were added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	False		

Indexes

Key	Name	Columns	Unique	File Group
	PK_FamilyNotes	FamilyNotesID	True	UData

Foreign Keys

Name	Columns
FK_FamilyNotes_Family	FamilyID->[dbo].[Family].[FamilyID]

SQL Script

```
CREATE TABLE [dbo].[FamilyNotes]
```

```
(
```

Author: liam

```
[FamilyNotesID] [int] NOT NULL IDENTITY(1, 1),
[FamilyID] [int] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_FamilyNotes_CreatedDate] DEFAULT
(getdate()),
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NOT NULL
) ON [UData]
GO
ALTER TABLE [dbo].[FamilyNotes] ADD CONSTRAINT [PK_FamilyNotes] PRIMARY KEY CLUSTERED
([FamilyNotesID]) ON [UData]
GO
ALTER TABLE [dbo].[FamilyNotes] ADD CONSTRAINT [FK_FamilyNotes_Family] FOREIGN KEY
([FamilyID]) REFERENCES [dbo].[Family] ([FamilyID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'table for Family notes', 'SCHEMA',
N'dbo', 'TABLE', N'FamilyNotes', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the notes where added', 'SCHEMA',
N'dbo', 'TABLE', N'FamilyNotes', 'COLUMN', N'CreatedDate'
GO
```

Uses

[dbo].[Family]

Used By

[dbo].[usp_InsertFamilyNotes]
[dbo].[usp_SLIInsertedDataSimplified]

[dbo].[FamilytoPhoneNumber]

MS_Description

linking table for Family and phonenumbers

Properties

Property	Value
File Group	UData
Row Count (~)	51
Created	5:47:40 PM Saturday, April 4, 2015
Last Modified	7:07:13 PM Saturday, April 4, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	FamilyID	int	4	False	
	PhoneNumberID	int	4	False	
	NumberPriority order which this number should be used to contact the Family (1 being first, 2 being 2nd . . .)	tinyint	1	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_FamilytoPhoneNumber	FamilyID, Phone-NumberID	True	UData

Foreign Keys

Name	Columns
FK_FamilytoPhoneNumber_Family	FamilyID->[dbo].[Family].[FamilyID]
FK_FamilytoPhoneNumber_PhoneNumber	PhoneNumberID->[dbo].[PhoneNumber].[PhoneNumberID]

SQL Script

```
CREATE TABLE [dbo].[FamilytoPhoneNumber]
(
    [FamilyID] [int] NOT NULL,
    [PhoneNumberID] [int] NOT NULL,
    [NumberPriority] [tinyint] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_FamilytoPhoneNumber_CreatedDate] DEFAULT
    (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[FamilytoPhoneNumber] ADD CONSTRAINT [PK_FamilytoPhoneNumber] PRIMARY
KEY CLUSTERED ([FamilyID], [PhoneNumberID]) ON [UData]
GO
ALTER TABLE [dbo].[FamilytoPhoneNumber] ADD CONSTRAINT [FK_FamilytoPhoneNumber_Family]
FOREIGN KEY ([FamilyID]) REFERENCES [dbo].[Family] ([FamilyID])
GO
ALTER TABLE [dbo].[FamilytoPhoneNumber] ADD CONSTRAINT [FK_FamilytoPhoneNumber_Phone-
Number] FOREIGN KEY ([PhoneNumberID]) REFERENCES [dbo].[PhoneNumber] ([PhoneNumberID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for Family and
phonenumbers', 'SCHEMA', N'dbo', 'TABLE', N'FamilytoPhoneNumber', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'order which this number should be used
to contact the Family (1 being first, 2 being 2nd . . . )', 'SCHEMA', N'dbo', 'TABLE',
N'FamilytoPhoneNumber', 'COLUMN', N'NumberPriority'
GO
```

Uses

[dbo].[Family]
[dbo].[PhoneNumber]

Used By

[dbo].[usp_InsertFamilytoPhoneNumber]
[dbo].[udf_SIFamilyPhoneNumber]

[dbo].[FamilytoProperty]	

MS_Description

linking table for Family and property - indicating when a Family occupied a property

Properties

Property	Value
File Group	UData
Row Count (~)	2024
Created	5:59:48 PM Saturday, April 11, 2015
Last Modified	8:17:29 PM Thursday, April 16, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	FamilytoPropertyID	int	4	False	1 - 1	
	FamilyID Primary family id mainly from legacy system	int	4	False		
	PropertyID	int	4	False		
	PropertyLinkTypeID	tinyint	1	True		
	ReviewStatusID	tinyint	1	True		
	StartDate date the Family started occupying the property	date	3	True		(getdate())
	EndDate date the Family stopped occupying the property	date	3	True		
	isPrimaryResidence	bit	1	True		
	CreatedDate	datetime	8	True		(getdate())
	ModifiedDate	datetime	8	True		

Indexes

Key	Name	Columns	Unique	File Group
	PK_FamilytoProperty	FamilytoPropertyID	True	UData

Author: liam

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateFamilytoProperty	True	True	After Update

Foreign Keys

Name	No Check	Columns
FK_FamilytoProperty_Family	True	FamilyID->[dbo].[Family].[FamilyID]
FK_FamilytoProperty_Property	True	PropertyID->[dbo].[Property].[PropertyID]
FK_FamilytoProperty_PropertyLinkType	True	PropertyLinkTypeID->[dbo].[PropertyLinkType].[PropertyLinkTypeID]
FK_FamilytoProperty_ReviewStatus	True	ReviewStatusID->[dbo].[ReviewStatus].[ReviewStatus-ID]

SQL Script

```

CREATE TABLE [dbo].[FamilytoProperty]
(
    [FamilytoPropertyID] [int] NOT NULL IDENTITY(1, 1),
    [FamilyID] [int] NOT NULL,
    [PropertyID] [int] NOT NULL,
    [PropertyLinkTypeID] [tinyint] NULL,
    [ReviewStatusID] [tinyint] NULL,
    [StartDate] [date] NULL CONSTRAINT [DF_FamilytoProperty_StartDate] DEFAULT (getdate()),
    [EndDate] [date] NULL,
    [isPrimaryResidence] [bit] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_FamilytoProperty_CreatedDate] DEFAULT
    (getdate()),
    [ModifiedDate] [datetime] NULL
) ON [UData]
GO

create trigger [dbo].[trUpdateFamilytoProperty] on [dbo].[FamilytoProperty] AFTER
UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update FamilytoProperty set ModifiedDate = getdate()
    where FamilytoPropertyID in (select FamilytoPropertyID from inserted)

    end
GO
ALTER TABLE [dbo].[FamilytoProperty] ADD CONSTRAINT [PK_FamilytoProperty] PRIMARY KEY
CLUSTERED ([FamilytoPropertyID]) ON [UData]
GO
ALTER TABLE [dbo].[FamilytoProperty] WITH NOCHECK ADD CONSTRAINT [FK_FamilytoProperty_-
Family] FOREIGN KEY ([FamilyID]) REFERENCES [dbo].[Family] ([FamilyID])

```

```
GO
ALTER TABLE [dbo].[FamilytoProperty] WITH NOCHECK ADD CONSTRAINT [FK_FamilytoProperty_Property] FOREIGN KEY ([PropertyID]) REFERENCES [dbo].[Property] ([PropertyID])
GO
ALTER TABLE [dbo].[FamilytoProperty] WITH NOCHECK ADD CONSTRAINT [FK_FamilytoProperty_PropertyLinkType] FOREIGN KEY ([PropertyLinkTypeID]) REFERENCES [dbo].[PropertyLinkType] ([PropertyLinkTypeID])
GO
ALTER TABLE [dbo].[FamilytoProperty] WITH NOCHECK ADD CONSTRAINT [FK_FamilytoProperty_ReviewStatus] FOREIGN KEY ([ReviewStatusID]) REFERENCES [dbo].[ReviewStatus] ([ReviewStatusID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for Family and property - indicating when a Family occupied a property', 'SCHEMA', N'dbo', 'TABLE', N'FamilytoProperty', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the Family stopped occupying the property', 'SCHEMA', N'dbo', 'TABLE', N'FamilytoProperty', 'COLUMN', N'EndDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Primary family id mainly from legacy system', 'SCHEMA', N'dbo', 'TABLE', N'FamilytoProperty', 'COLUMN', N'FamilyID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the Family started occupying the property', 'SCHEMA', N'dbo', 'TABLE', N'FamilytoProperty', 'COLUMN', N'StartDate'
GO
```

Uses

[dbo].[Family]
[dbo].[Property]
[dbo].[PropertyLinkType]
[dbo].[ReviewStatus]

Used By

[dbo].[usp_InsertFamilytoProperty]

[dbo].[FileType]	

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	4
Created	6:03:29 PM Friday, January 2, 2015
Last Modified	4:55:07 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	FileTypeID	smallint	2	False	1 - 1	
	FileTypeName	varchar(50)	50	False		
	FileTypeDescription	varchar(253)	253	True		
	CreatedDate	datetime	8	True		(getdate())
	ModifiedDate	datetime	8	True		

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_FileTypes	FileTypeID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateFileType	True	True	After Update

SQL Script

```
CREATE TABLE [dbo].[FileType]
(
[FileTypeID] [smallint] NOT NULL IDENTITY(1, 1),
[FileName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
[FileTypeDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_FileType_CreatedDate] DEFAULT (getdate()),
[ModifiedDate] [datetime] NULL
```

Project> (local)> User databases> LCCHPDev> Tables> dbo.FileType

```
) ON [UData]
GO
create trigger [dbo].[trUpdateFileType] on [dbo].[FileType] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update FileType set ModifiedDate = getdate() where
FileTypeID in (select FileTypeID from inserted)

    end
GO
ALTER TABLE [dbo].[FileType] ADD CONSTRAINT [PK_FileTypes] PRIMARY KEY CLUSTERED
([FileTypeID]) ON [UData]
GO
```

Author: liam

 [dbo].[Flag]	

MS_Description

Collection of flag information

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	0
Created	3:00:25 PM Saturday, April 11, 2015
Last Modified	3:00:25 PM Saturday, April 11, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	FlagID	tinyint	1	False	1 - 1	
	FlagDescription Detailed description of the flag	varchar(253)	253	True		
	FlagName short name for the flag	varchar(50)	50	True		
	HistoricFlagID historic flg ID from access database	char(1)	1	True		
	ModifiedDate last modified date for the record	datetime	8	True		
	CreatedDate date the record was created	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_Flag	FlagID	True	UData

SQL Script

```
CREATE TABLE [dbo].[Flag]
(
[FlagID] [tinyint] NOT NULL IDENTITY(1, 1),
[FlagDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
```

Author: liam

Project> (local)> User databases> LCCHPDev> Tables> dbo.Flag

```
[FlagName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[HistoricFlagID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Flag_CreatedDate] DEFAULT (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[Flag] ADD CONSTRAINT [PK_Flag] PRIMARY KEY CLUSTERED ([FlagID]) ON
[UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Collection of flag information',
'SCHEMA', N'dbo', 'TABLE', N'Flag', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the record was created',
'SCHEMA', N'dbo', 'TABLE', N'Flag', 'COLUMN', N'CreatedDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Detailed description of the flag',
'SCHEMA', N'dbo', 'TABLE', N'Flag', 'COLUMN', N'FlagDescription'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'short name for the flag', 'SCHEMA',
N'dbo', 'TABLE', N'Flag', 'COLUMN', N'FlagName'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'historic flg ID from access database',
'SCHEMA', N'dbo', 'TABLE', N'Flag', 'COLUMN', N'HistoricFlagID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'last modified date for the record',
'SCHEMA', N'dbo', 'TABLE', N'Flag', 'COLUMN', N'ModifiedDate'
GO
```

[dbo].[ForeignFood]

MS_Description

collection of various foreign foods

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	2
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	4:55:07 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	ForeignFoodID	int	4	False	1 - 1	
	ForeignFoodName	varchar(50)	50	True		
	ForeignFoodDescription	varchar(253)	253	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_ForeignFood	ForeignFoodID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateForeignFood	True	True	After Update

SQL Script

```
CREATE TABLE [dbo].[ForeignFood]
(
[ForeignFoodID] [int] NOT NULL IDENTITY(1, 1),
[ForeignFoodName] [varchar](50) COLLATE SQL_Latin1_General_CI_AS NULL,
```

Author: liam

```
[ForeignFoodDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_ForeignFood_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
create trigger [dbo].[trUpdateForeignFood] on [dbo].[ForeignFood] AFTER UPDATE
as
begin
if @@rowcount = 0
return
if not update(ModifiedDate) update ForeignFood set ModifiedDate = getdate() where
ForeignFoodID in (select ForeignFoodID from inserted)

end
GO
ALTER TABLE [dbo].[ForeignFood] ADD CONSTRAINT [PK_ForeignFood] PRIMARY KEY CLUSTERED
([ForeignFoodID]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of various foreign foods',
'SCHEMA', N'dbo', 'TABLE', N'ForeignFood', NULL, NULL
GO
```

Used By

[dbo].[ForeignFoodtoCountry]
[dbo].[PersontoForeignFood]
[dbo].[usp_InsertForeignFood]

[dbo].[ForeignFoodtoCountry]

MS_Description

foreign food and country linking table

Properties

Property	Value
File Group	UData
Row Count (~)	2
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	2:10:04 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	ForeignFoodID	int	4	False	
	CountryID	tinyint	1	False	
	ModifiedDate	datetime	8	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_ForeignFoodtoCountry	ForeignFoodID, CountryID	True	UData

Foreign Keys

Name	Columns
FK_ForeignFoodtoCountry_Country	CountryID->[dbo].[Country].[CountryID]
FK_ForeignFoodtoCountry_ForeignFood	ForeignFoodID->[dbo].[ForeignFood].[ForeignFoodID]

SQL Script

```
CREATE TABLE [dbo].[ForeignFoodtoCountry]
(
    [ForeignFoodID] [int] NOT NULL,
```

Author: liam

```
[CountryID] [tinyint] NOT NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_ForeignFoodtoCountry_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[ForeignFoodtoCountry] ADD CONSTRAINT [PK_ForeignFoodtoCountry]
PRIMARY KEY CLUSTERED ([ForeignFoodID], [CountryID]) ON [UData]
GO
ALTER TABLE [dbo].[ForeignFoodtoCountry] ADD CONSTRAINT [FK_ForeignFoodtoCountry_-
Country] FOREIGN KEY ([CountryID]) REFERENCES [dbo].[Country] ([CountryID])
GO
ALTER TABLE [dbo].[ForeignFoodtoCountry] ADD CONSTRAINT [FK_ForeignFoodtoCountry_-
ForeignFood] FOREIGN KEY ([ForeignFoodID]) REFERENCES [dbo].[ForeignFood] ([ForeignFood-
ID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'foreign food and country linking
table', 'SCHEMA', N'dbo', 'TABLE', N'ForeignFoodtoCountry', NULL, NULL
GO
```

Uses

[dbo].[Country]
[dbo].[ForeignFood]

Used By

[dbo].[usp_InsertForeignFoodtoCountry]

[dbo].[Frequency]

MS_Description

Collection of frequencies

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	0
Created	3:44:11 PM Saturday, April 11, 2015
Last Modified	3:44:11 PM Saturday, April 11, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	FrequencyID	tinyint	1	False	1 - 1	
	FrequencyDescription Detailed description of the frequency type	varchar(253)	253	True		
	FrequencyName short name for the frequency type	varchar(50)	50	True		
	HistoricFrequencyID historic frequency ID from access database	char(1)	1	True		
	ModifiedDate last modified date for the record	datetime	8	True		
	CreatedDate date the record was created	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_Frequency	FrequencyID	True	UData

SQL Script

```
CREATE TABLE [dbo].[Frequency]
(

```

Author: liam

Project> (local)> User databases> LCCHPDev> Tables> dbo.Frequency

```
[FrequencyID] [tinyint] NOT NULL IDENTITY(1, 1),
[FrequencyDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[FrequencyName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[HistoricFrequencyID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Frequency_CreatedDate] DEFAULT (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[Frequency] ADD CONSTRAINT [PK_Frequency] PRIMARY KEY CLUSTERED
([FrequencyID]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Collection of frequencies', 'SCHEMA',
N'dbo', 'TABLE', N'Frequency', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the record was created',
'SCHEMA', N'dbo', 'TABLE', N'Frequency', 'COLUMN', N'CreatedDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Detailed description of the frequency
type', 'SCHEMA', N'dbo', 'TABLE', N'Frequency', 'COLUMN', N'FrequencyDescription'
GO
EXEC sp_addextendedproperty N'MS_Description', N'short name for the frequency type',
'SCHEMA', N'dbo', 'TABLE', N'Frequency', 'COLUMN', N'FrequencyName'
GO
EXEC sp_addextendedproperty N'MS_Description', N'historic frequency ID from access
database', 'SCHEMA', N'dbo', 'TABLE', N'Frequency', 'COLUMN', N'HistoricFrequencyID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'last modified date for the record',
'SCHEMA', N'dbo', 'TABLE', N'Frequency', 'COLUMN', N'ModifiedDate'
GO
```

[dbo].[GiftCard]

MS_Description

collection of gift certificate objects

Properties

Property	Value
File Group	UData
Row Count (~)	8
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	4:55:07 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	GiftCardID unique identifier for the gift certificate	int	4	False	1 - 1	
	GiftCardValue value of the gift certificate	money	8	False		((25))
	IssueDate	date	3	False		(getdate())
	PersonID	int	4	False		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_GiftCertificate	GiftCardID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateGiftCard	True	True	After Update

Foreign Keys

Name	Columns
FK_GiftCard_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```

CREATE TABLE [dbo].[GiftCard]
(
    [GiftCardID] [int] NOT NULL IDENTITY(1, 1),
    [GiftCardValue] [money] NOT NULL CONSTRAINT [DF_GiftCertificate_GiftCertificateValue]
        DEFAULT ((25)),
    [IssueDate] [date] NOT NULL CONSTRAINT [DF_GiftCard_IssueDate] DEFAULT (getdate()),
    [PersonID] [int] NOT NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_GiftCard_CreatedDate] DEFAULT (getdate())
) ON [UData]
GO
create trigger [dbo].[trUpdateGiftCard] on [dbo].[GiftCard] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update GiftCard set ModifiedDate = getdate() where
    GiftCardID in (select GiftCardID from inserted)

    end
GO
ALTER TABLE [dbo].[GiftCard] ADD CONSTRAINT [PK_GiftCertificate] PRIMARY KEY CLUSTERED
([GiftCardID]) ON [UData]
GO
ALTER TABLE [dbo].[GiftCard] ADD CONSTRAINT [FK_GiftCard_Person] FOREIGN KEY ([PersonID])
    REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of gift certificate
objects', 'SCHEMA', N'dbo', 'TABLE', N'GiftCard', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'unique identifier for the gift
certificate', 'SCHEMA', N'dbo', 'TABLE', N'GiftCard', 'COLUMN', N'GiftCardID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'value of the gift certificate',
'SCHEMA', N'dbo', 'TABLE', N'GiftCard', 'COLUMN', N'GiftCardValue'
GO

```

Uses

[dbo].[Person]

Used By

[dbo].[usp_InsertGiftCard]

Author: liam

[dbo].[HistoricContribution]

MS_Description

Collection of historic contribution classifications

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	0
Created	4:28:26 PM Saturday, April 11, 2015
Last Modified	4:28:26 PM Saturday, April 11, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	HistoricContributionID	tinyint	1	False	1 - 1	
	HistoricContributionDescription Detailed description of the historic contribution	varchar(253)	253	True		
	HistoricContributionName short name for the historic contribution name	varchar(50)	50	True		
	HistoricHistoricContributionID historic historic contribution ID from access database	char(1)	1	True		
	ModifiedDate last modified date for the record	datetime	8	True		
	CreatedDate date the record was created	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_HistoricContribution	HistoricContributionID	True	UData

SQL Script

```
CREATE TABLE [dbo].[HistoricContribution]
```

```

(
[HistoricContributionID] [tinyint] NOT NULL IDENTITY(1, 1),
[HistoricContributionDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS
NULL,
[HistoricContributionName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[HistoricHistoricContributionID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_HistoricContribution_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[HistoricContribution] ADD CONSTRAINT [PK_HistoricContribution]
PRIMARY KEY CLUSTERED ([HistoricContributionID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Collection of historic contribution
classifications', 'SCHEMA', N'dbo', 'TABLE', N'HistoricContribution', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the record was created',
'SCHEMA', N'dbo', 'TABLE', N'HistoricContribution', 'COLUMN', N'CreatedDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Detailed description of the historic
contribution', 'SCHEMA', N'dbo', 'TABLE', N'HistoricContribution', 'COLUMN', N'Historic-
ContributionDescription'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'short name for the historic
contribution name', 'SCHEMA', N'dbo', 'TABLE', N'HistoricContribution', 'COLUMN',
N'HistoricContributionName'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'historic historic contribution ID from
access database', 'SCHEMA', N'dbo', 'TABLE', N'HistoricContribution', 'COLUMN',
N'HistoricHistoricContributionID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'last modified date for the record',
'SCHEMA', N'dbo', 'TABLE', N'HistoricContribution', 'COLUMN', N'ModifiedDate'
GO

```

[dbo].[Hobby]

MS_Description

collection of hobbies

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	22
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	10:26:09 PM Thursday, February 12, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	HobbyID unique identifier of hobby objects	smallint	2	False	1 - 1	
	HobbyDescription short description of the hobby	varchar(253)	253	True		
	HobbyName	varchar(50)	50	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())
	LeadExposure	bit	1	True		

Indexes

Key	Name	Columns	Unique	File Group
	PK_Hobby	HobbyID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateHobby	True	True	After Update

SQL Script

```
CREATE TABLE [dbo].[Hobby]
(
[HobbyID] [smallint] NOT NULL IDENTITY(1, 1),
[HobbyDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[HobbyName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Hobby_CreatedDate] DEFAULT (getdate()),
[LeadExposure] [bit] NULL
) ON [UData]
GO
create trigger [dbo].[trUpdateHobby] on [dbo].[Hobby] AFTER UPDATE
as
begin
if @@rowcount = 0
    return
if not update(ModifiedDate) update Hobby set ModifiedDate = getdate() where Hobby-
ID in (select HobbyID from inserted)

end
GO
ALTER TABLE [dbo].[Hobby] ADD CONSTRAINT [PK_Hobby] PRIMARY KEY CLUSTERED ([HobbyID])
ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'collection of hobbies', 'SCHEMA',
N'dbo', 'TABLE', N'Hobby', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'short description of the hobby',
'SCHEMA', N'dbo', 'TABLE', N'Hobby', 'COLUMN', N'HobbyDescription'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'unique identifier of hobby objects',
'SCHEMA', N'dbo', 'TABLE', N'Hobby', 'COLUMN', N'HobbyID'
GO
```

Used By

[dbo].[PersontoHobby]
[dbo].[usp_InsertHobby]
[dbo].[usp_SIHobby]

[dbo].[HomeRemedy]

MS_Description

collection of home remedies

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	3
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	4:55:07 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	HomeRemedyID	int	4	False	1 - 1	
	HomeRemedyName	varchar(50)	50	False		
	HomeRemedyDescription	varchar(253)	253	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_HomeRemedies	HomeRemedyID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateHomeRemedy	True	True	After Update

SQL Script

```
CREATE TABLE [dbo].[HomeRemedy]
(
[HomeRemedyID] [int] NOT NULL IDENTITY(1, 1),
[HomeRemedyName] [varchar](50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
```

Author: liam

```
[HomeRemedyDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_HomeRemedy_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
create trigger [dbo].[trUpdateHomeRemedy] on [dbo].[HomeRemedy] AFTER UPDATE
as
begin
if @@rowcount = 0
return
if not update(ModifiedDate) update HomeRemedy set ModifiedDate = getdate() where
HomeRemedyID in (select HomeRemedyID from inserted)

end
GO
ALTER TABLE [dbo].[HomeRemedy] ADD CONSTRAINT [PK_HomeRemedies] PRIMARY KEY CLUSTERED
([HomeRemedyID]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of home remedies',
'SCHEMA', N'dbo', 'TABLE', N'HomeRemedy', NULL, NULL
GO
```

Used By

[dbo].[PersonstoHomeRemedy]
[dbo].[usp_InsertHomeRemedies]

[dbo].[HouseholdSourcesofLead]

MS_Description

household items that may contribute to EBL

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	2
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	4:55:07 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	HouseholdSourcesofLeadID	int	4	False	1 - 1	
	HouseholdItemName	varchar(50)	50	True		
	HouseholdItemDescription	varchar(253)	253	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_HouseholdSourcesofLead	HouseholdSourcesofLeadID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateHouseholdSourcesofLead	True	True	After Update

SQL Script

```
CREATE TABLE [dbo].[HouseholdSourcesofLead]
(
[HouseholdSourcesofLeadID] [int] NOT NULL IDENTITY(1, 1),
```

Author: liam

```
[HouseholdItemName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[HouseholdItemDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_HouseholdSourcesofLead_CreatedDate]
DEFAULT (getdate())
) ON [UData]
GO
create trigger [dbo].[trUpdateHouseholdSourcesofLead] on [dbo].[HouseholdSourcesofLead]
AFTER UPDATE
as
begin
if @@rowcount = 0
return
if not update(ModifiedDate) update HouseholdSourcesofLead set ModifiedDate =
getdate() where HouseholdSourcesofLeadID in (select HouseholdSourcesofLeadID from
inserted)

end
GO
ALTER TABLE [dbo].[HouseholdSourcesofLead] ADD CONSTRAINT [PK_HouseholdSourcesofLead]
PRIMARY KEY CLUSTERED ([HouseholdSourcesofLeadID]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'household items that may contribute to
EBL', 'SCHEMA', N'dbo', 'TABLE', N'HouseholdSourcesofLead', NULL, NULL
GO
```

Used By

[dbo].[PropertytoHouseholdSourcesofLead]
[dbo].[usp_InsertHouseholdSourcesofLead]

[dbo].[InsuranceProvider]

MS_Description

collection of insurance companies

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	6
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	4:55:07 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
PK_C	InsuranceProviderID unique identifier for insurance company	smallint	2	False	1 - 1	
	InsuranceProviderName name of the insurance company	varchar(50)	50	False		
	CreatedDate	datetime	8	True		(getdate())
	ModifiedDate	datetime	8	True		

Indexes

Key	Name	Columns	Unique	File Group
PK_C	PK_InsuranceProvider	InsuranceProviderID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateInsuranceProvider	True	True	After Update

SQL Script

```
CREATE TABLE [dbo].[InsuranceProvider]
```

Author: liam

```
(  
    [InsuranceProviderID] [smallint] NOT NULL IDENTITY(1, 1),  
    [InsuranceProviderName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,  
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_InsuranceProvider_CreatedDate] DEFAULT  
        (getdate()),  
    [ModifiedDate] [datetime] NULL  
) ON [UData]  
GO  
create trigger [dbo].[trUpdateInsuranceProvider] on [dbo].[InsuranceProvider] AFTER  
UPDATE  
as  
begin  
    if @@rowcount = 0  
        return  
    if not update(ModifiedDate) update InsuranceProvider set ModifiedDate = getdate()  
    where InsuranceProviderID in (select InsuranceProviderID from inserted)  
  
end  
GO  
ALTER TABLE [dbo].[InsuranceProvider] ADD CONSTRAINT [PK_InsuranceProvider] PRIMARY KEY  
CLUSTERED ([InsuranceProviderID]) ON [UData]  
GO  
EXEC sp_addextendedproperty N'MS_Description', N'collection of insurance companies',  
    'SCHEMA', N'dbo', 'TABLE', N'InsuranceProvider', NULL, NULL  
GO  
EXEC sp_addextendedproperty N'MS_Description', N'unique identifier for insurance  
company', 'SCHEMA', N'dbo', 'TABLE', N'InsuranceProvider', 'COLUMN', N'Insurance-  
ProviderID'  
GO  
EXEC sp_addextendedproperty N'MS_Description', N'name of the insurance company',  
    'SCHEMA', N'dbo', 'TABLE', N'InsuranceProvider', 'COLUMN', N'InsuranceProviderName'  
GO
```

Used By

[dbo].[PersontoInsurance]
[dbo].[usp_InsertInsuranceProvider]

[dbo].[Lab]

MS_Description

collection of lab names and basic attributes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	10
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	8:24:48 PM Wednesday, March 4, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	LabID unique identifier for the lab object	int	4	False	1 - 1	
	LabName	varchar(50)	50	True		
	LabDescription	varchar(253)	253	True		
	CreatedDate	datetime	8	True		(getdate())
	ModifiedDate	datetime	8	True		

Indexes

Key	Name	Columns	Unique	File Group
	PK_Lab	LabID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateLab	True	True	After Update

SQL Script

```

CREATE TABLE [dbo].[Lab]
(
    [LabID] [int] NOT NULL IDENTITY(1, 1),
    [LabName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
    [LabDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_Lab_CreatedDate] DEFAULT (getdate()),
    [ModifiedDate] [datetime] NULL
) ON [UData]
GO
create trigger [dbo].[trUpdateLab] on [dbo].[Lab] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update Lab set ModifiedDate = getdate() where LabID
    in (select LabID from inserted)

end
GO
ALTER TABLE [dbo].[Lab] ADD CONSTRAINT [PK_Lab] PRIMARY KEY CLUSTERED ([LabID]) ON
[UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of lab names and basic
attributes', 'SCHEMA', N'dbo', 'TABLE', N'Lab', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'unique identifier for the lab object',
'SCHEMA', N'dbo', 'TABLE', N'Lab', 'COLUMN', N'LabID'
GO

```

Used By

- [dbo].[BloodTestResults]
- [dbo].[LabNotes]
- [dbo].[MediumSampleResults]
- [dbo].[usp_InsertLab]
- [dbo].[usp_SILabName]

[dbo].[LabNotes]	

MS_Description

linking table for access agreement and access agreement notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	1
Created	12:21:25 AM Tuesday, February 17, 2015
Last Modified	12:21:25 AM Tuesday, February 17, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	LabNotesID	int	4	False	1 - 1	
	LabID	int	4	False		
	CreatedDate date the notes were added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	False		

Indexes

Key	Name	Columns	Unique	File Group
	PK_LabNotes	LabNotesID	True	UData

Foreign Keys

Name	Columns
FK_LabNotes_Lab	LabID->[dbo].[Lab].[LabID]

SQL Script

```
CREATE TABLE [dbo].[LabNotes]
(
```

Author: liam

```
[LabNotesID] [int] NOT NULL IDENTITY(1, 1),
[LabID] [int] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_LabNotes_CreatedDate] DEFAULT (getdate()),
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NOT NULL
) ON [UData]
GO
ALTER TABLE [dbo].[LabNotes] ADD CONSTRAINT [PK_LabNotes] PRIMARY KEY CLUSTERED ([Lab-
NotesID]) ON [UData]
GO
ALTER TABLE [dbo].[LabNotes] ADD CONSTRAINT [FK_LabNotes_Lab] FOREIGN KEY ([LabID])
REFERENCES [dbo].[Lab] ([LabID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for access agreement and
access agreement notes', 'SCHEMA', N'dbo', 'TABLE', N'LabNotes', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the notes where added', 'SCHEMA',
N'dbo', 'TABLE', N'LabNotes', 'COLUMN', N'CreatedDate'
GO
```

Uses

[dbo].[Lab]

Used By

[dbo].[usp_InsertLabNotes]

[dbo].[Language]

MS_Description

collection of spoken languages

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	26
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	6:08:22 PM Thursday, April 16, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
PK_C	LanguageID unique identifier of language object	tinyint	1	False	1 - 1	
	LanguageName spoken language	varchar(50)	50	False		
	HistoricPrimaryLanguageCode	char(1)	1	True		
	CreatedDate	datetime	8	True		(getdate())
	ModifiedDate	datetime	8	True		

Indexes

Key	Name	Columns	Unique	File Group
PK_C	PK_Language	LanguageID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateLanguage	True	True	After Update

SQL Script

```

CREATE TABLE [dbo].[Language]
(
[LanguageID] [tinyint] NOT NULL IDENTITY(1, 1),
[LanguageName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
[HistoricPrimaryLanguageCode] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Language_CreatedDate] DEFAULT (getdate()),
[ModifiedDate] [datetime] NULL
) ON [UData]
GO
create trigger [dbo].[trUpdateLanguage] on [dbo].[Language] AFTER UPDATE
as
begin
if @@rowcount = 0
return
if not update(ModifiedDate) update Language set ModifiedDate = getdate() where
LanguageID in (select LanguageID from inserted)

end
GO
ALTER TABLE [dbo].[Language] ADD CONSTRAINT [PK_Language] PRIMARY KEY CLUSTERED
([LanguageID]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of spoken languages',
'SCHEMA', N'dbo', 'TABLE', N'Language', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'unqiue identifier of langauge object',
'SCHEMA', N'dbo', 'TABLE', N'Language', 'COLUMN', N'LanguageID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'spoken language', 'SCHEMA', N'dbo',
'TABLE', N'Language', 'COLUMN', N'LanguageName'
GO

```

Used By

[dbo].[PersontoLanguage]
[dbo].[usp_InsertLanguage]

[dbo].[LCCHPAttachments]

Properties

Property	Value
File Group	UData
Filestream File Group	LCCHPAttachments
Heap	True
Row Count (~)	4
Created	8:36:33 PM Friday, January 2, 2015
Last Modified	8:36:33 PM Friday, January 2, 2015

SQL Script

```
CREATE TABLE [dbo].[LCCHPAttachments] AS FILETABLE ON [UData] FILESTREAM_ON
[LCCHPAttachments]
WITH
(
    FILETABLE_DIRECTORY = N'LCCHPAttachmentsDev', FILETABLE_COLLATE_FILENAME = SQL_-
Latin1_General_CI_AS
)
GO
```

[dbo].[Medium]

MS_Description

collection of mediums that are tested

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	6
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	4:55:07 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	MediumID unique identifier of the medium	int	4	False	1 - 1	
	MediumName short name for the medium	varchar(50)	50	False		
	MediumDescription short description of the medium	varchar(253)	253	True		
	TriggerLevel	int	4	True		
	TriggerLevelUnitsID	int	4	True		
	HistoricMediumCode mediumcode identifier from legacy database	char(1)	1	True		
	CreatedDate	datetime	8	True		(getdate())
	ModifiedDate	datetime	8	True		

Indexes

Key	Name	Columns	Unique	File Group
	PK_Medium	MediumID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateMedium	True	True	After Update

SQL Script

```

CREATE TABLE [dbo].[Medium]
(
[MediumID] [int] NOT NULL IDENTITY(1, 1),
[MediumName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
[MediumDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[TriggerLevel] [int] NULL,
[TriggerLevelUnitsID] [int] NULL,
[HistoricMediumCode] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Medium_CreatedDate] DEFAULT (getdate()),
[ModifiedDate] [datetime] NULL
) ON [UData]
GO
create trigger [dbo].[trUpdateMedium] on [dbo].[Medium] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update Medium set ModifiedDate = getdate() where
MediumID in (select MediumID from inserted)

    end
GO
ALTER TABLE [dbo].[Medium] ADD CONSTRAINT [PK_Medium] PRIMARY KEY CLUSTERED ([Medium-
ID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'collection of mediums that are
tested', 'SCHEMA', N'dbo', 'TABLE', N'Medium', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'mediumcode identifier from legacy
database', 'SCHEMA', N'dbo', 'TABLE', N'Medium', 'COLUMN', N'HistoricMediumCode'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'short description of the medium',
'SCHEMA', N'dbo', 'TABLE', N'Medium', 'COLUMN', N'MediumDescription'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'unique identifier of the medium',
'SCHEMA', N'dbo', 'TABLE', N'Medium', 'COLUMN', N'MediumID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'short name for the medium', 'SCHEMA',
N'dbo', 'TABLE', N'Medium', 'COLUMN', N'MediumName'
GO

```

Used By

[dbo].[MediumSampleResults]

Author: liam

Project> (local)> User databases> LCCHPDev> Tables> dbo.Medium

[dbo].[PropertytoMedium]
[dbo].[usp_InsertMedium]
[dbo].[usp_InsertMediumSampleResults]

Author: liam

[dbo].[MediumSampleResults]

MS_Description

collection of test results for various mediums

Properties

Property	Value
File Group	UData
Row Count (~)	12
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	7:00:52 PM Thursday, April 16, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	MediumSampleResultsID	int	4	False	1 - 1	
	MediumID tested medium id	int	4	False		
	MediumSampleValue value of the test result for the medium	numeric(9,4)	5	True		
	SampleLevelCategoryID sample level category	tinyint	1	True		
	MediumSampleDate date the medium was tested	date	3	False		(getdate())
	LabID id of the lab to which the sample was submitted	int	4	True		
	LabSubmissionDate date the sample was submitted to the lab	date	3	True		
	IsAboveTriggerLevel	bit	1	True		
	UnitsID	smallint	2	True		
	CreatedDate	datetime	8	True		(getdate())
	ModifiedDate	datetime	8	True		

Indexes

Key	Name	Columns	Unique	File Group
 PK_MediumTestResults		MediumSampleResults-ID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateMediumSampleResults	True	True	After Update

Check Constraints

Name	On Column	Constraint
ck_MediumSampleResults_MediumSampleDate	MediumSampleDate	([dbo].[udf_DateInThe-Past]([MediumSampleDate])=(1))

Foreign Keys

Name	Columns
FK_MediumSampleResults_Lab	LabID->[dbo].[Lab].[LabID]
FK_MediumSampleResults_Medium	MediumID->[dbo].[Medium].[MediumID]
FK_MediumSampleResults_SampleLevelCategory	SampleLevelCategoryID->[dbo].[SampleLevel-Category].[SampleLevelCategoryID]
FK_MediumSampleResults_Units	UnitsID->[dbo].[Units].[UnitsID]

SQL Script

```

CREATE TABLE [dbo].[MediumSampleResults]
(
    [MediumSampleResultsID] [int] NOT NULL IDENTITY(1, 1),
    [MediumID] [int] NOT NULL,
    [MediumSampleValue] [numeric] (9, 4) NULL,
    [SampleLevelCategoryID] [tinyint] NULL,
    [MediumSampleDate] [date] NOT NULL CONSTRAINT [DF_MediumTestResults_MediumTestDate]
        DEFAULT (getdate()),
    [LabID] [int] NULL,
    [LabSubmissionDate] [date] NULL,
    [IsAboveTriggerLevel] [bit] NULL,
    [UnitsID] [smallint] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_MediumSampleResults_CreatedDate] DEFAULT
        (getdate()),
    [ModifiedDate] [datetime] NULL
) ON [UData]
GO
create trigger [dbo].[trUpdateMediumSampleResults] on [dbo].[MediumSampleResults] AFTER
UPDATE

```

```

as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update MediumSampleResults set ModifiedDate =
getdate() where MediumSampleResultsID in (select MediumSampleResultsID from inserted)

end
GO
ALTER TABLE [dbo].[MediumSampleResults] ADD CONSTRAINT [ck_MediumSampleResults_Medium-
SampleDate] CHECK (([dbo].[udf_DateInThePast]([MediumSampleDate])=(1)))
GO
ALTER TABLE [dbo].[MediumSampleResults] ADD CONSTRAINT [PK_MediumTestResults] PRIMARY
KEY CLUSTERED ([MediumSampleResultsID]) ON [UData]
GO
ALTER TABLE [dbo].[MediumSampleResults] ADD CONSTRAINT [FK_MediumSampleResults_Lab]
FOREIGN KEY ([LabID]) REFERENCES [dbo].[Lab] ([LabID])
GO
ALTER TABLE [dbo].[MediumSampleResults] ADD CONSTRAINT [FK_MediumSampleResults_Medium]
FOREIGN KEY ([MediumID]) REFERENCES [dbo].[Medium] ([MediumID])
GO
ALTER TABLE [dbo].[MediumSampleResults] ADD CONSTRAINT [FK_MediumSampleResults_Sample-
LevelCategory] FOREIGN KEY ([SampleLevelCategoryID]) REFERENCES [dbo].[SampleLevel-
Category] ([SampleLevelCategoryID])
GO
ALTER TABLE [dbo].[MediumSampleResults] ADD CONSTRAINT [FK_MediumSampleResults_Units]
FOREIGN KEY ([UnitsID]) REFERENCES [dbo].[Units] ([UnitsID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of test results for various
mediums', 'SCHEMA', 'dbo', 'TABLE', 'N'MediumSampleResults', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the lab to which the sample was
submitted', 'SCHEMA', 'dbo', 'TABLE', 'N'MediumSampleResults', 'COLUMN', 'N'LabID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the sample was submitted to the
lab', 'SCHEMA', 'dbo', 'TABLE', 'N'MediumSampleResults', 'COLUMN', N'LabSubmissionDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'tested medium id', 'SCHEMA', 'dbo',
'TABLE', 'N'MediumSampleResults', 'COLUMN', N'MediumID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the medium was tested', 'SCHEMA',
N'dbo', 'TABLE', 'N'MediumSampleResults', 'COLUMN', N'MediumSampleDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'value of the test result for the
medium', 'SCHEMA', 'dbo', 'TABLE', 'N'MediumSampleResults', 'COLUMN', N'MediumSample-
Value'
GO
EXEC sp_addextendedproperty N'MS_Description', N'sample level category', 'SCHEMA',
N'dbo', 'TABLE', 'N'MediumSampleResults', 'COLUMN', N'SampleLevelCategoryID'
GO

```

Uses

[dbo].[Lab]
 [dbo].[Medium]

Author: liam

[dbo].[SampleLevelCategory]

[dbo].[Units]

[dbo].[udf_DateInThePast]

Used By

[dbo].[MediumSampleResultsNotes]

[dbo].[usp_InsertMediumSampleResults]

[dbo].[MediumSampleResultsNotes]

MS_Description

linking table for access agreement and access agreement notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	2
Created	7:00:52 PM Thursday, April 16, 2015
Last Modified	7:00:52 PM Thursday, April 16, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	MediumSampleResultsNotesID	int	4	False	1 - 1	
 D	MediumSampleResultsID	int	4	False		
	CreatedDate date the notes where added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	True		

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_MediumSampleResultsNotes	MediumSample- ResultsNotesID	True	UData

Foreign Keys

Name	Columns
FK_MediumSampleResultsNotes_MediumSampleResults	MediumSampleResultsID->[dbo].[MediumSample- Results].[MediumSampleResultsID]

SQL Script

```
CREATE TABLE [dbo].[MediumSampleResultsNotes]
(
[MediumSampleResultsNotesID] [int] NOT NULL IDENTITY(1, 1),
[MediumSampleResultsID] [int] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_MediumSampleResultsNotes_CreatedDate]
DEFAULT (getdate()),
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NULL
) ON [UData]
GO
ALTER TABLE [dbo].[MediumSampleResultsNotes] ADD CONSTRAINT [PK_MediumSampleResults-
Notes] PRIMARY KEY CLUSTERED ([MediumSampleResultsNotesID]) ON [UData]
GO
ALTER TABLE [dbo].[MediumSampleResultsNotes] ADD CONSTRAINT [FK_MediumSampleResults-
Notes_MediumSampleResults] FOREIGN KEY ([MediumSampleResultsID]) REFERENCES
[dbo].[MediumSampleResults] ([MediumSampleResultsID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for access agreement and
access agreement notes', 'SCHEMA', N'dbo', 'TABLE', N'MediumSampleResultsNotes', NULL,
NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the notes where added', 'SCHEMA',
N'dbo', 'TABLE', N'MediumSampleResultsNotes', 'COLUMN', N'CreatedDate'
GO
```

Uses

[dbo].[MediumSampleResults]

Used By

[dbo].[usp_InsertMediumSampleResultsNotes]

[dbo].[Method]

MS_Description

Collection of method classifications

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	0
Created	4:43:56 PM Saturday, April 11, 2015
Last Modified	4:43:56 PM Saturday, April 11, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	MethodID	tinyint	1	False	1 - 1	
	MethodDescription Detailed description of the method	varchar(253)	253	True		
	MethodName short name for the method	varchar(50)	50	True		
	HistoricMethodID historic method ID from access database	char(1)	1	True		
	ModifiedDate last modified date for the record	datetime	8	True		
	CreatedDate date the record was created	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_Method	MethodID	True	UData

SQL Script

```
CREATE TABLE [dbo].[Method]
(
[MethodID] [tinyint] NOT NULL IDENTITY(1, 1),
```

Author: liam

Project> (local)> User databases> LCCHPDev> Tables> dbo.Method

```
[MethodDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[MethodName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[HistoricMethodID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Method_CreatedDate] DEFAULT (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[Method] ADD CONSTRAINT [PK_Method] PRIMARY KEY CLUSTERED ([MethodID])
ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Collection of method classifications',
'SCHEMA', N'dbo', 'TABLE', N'Method', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the record was created',
'SCHEMA', N'dbo', 'TABLE', N'Method', 'COLUMN', N'CreatedDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'historic method ID from access
database', 'SCHEMA', N'dbo', 'TABLE', N'Method', 'COLUMN', N'HistoricMethodID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Detailed description of the method',
'SCHEMA', N'dbo', 'TABLE', N'Method', 'COLUMN', N'MethodDescription'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'short name for the method', 'SCHEMA',
N'dbo', 'TABLE', N'Method', 'COLUMN', N'MethodName'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'last modified date for the record',
'SCHEMA', N'dbo', 'TABLE', N'Method', 'COLUMN', N'ModifiedDate'
GO
```

[dbo].[Occupation]	

MS_Description

collection of occupation objects

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	21
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	1:03:20 AM Tuesday, February 17, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	OccupationID unique identifier of the occupation	int	4	False	1 - 1	
	OccupationName name of the occupation	varchar(50)	50	False		
	OccupationDescription short description of the occupation	varchar(253)	253	True		
	CreatedDate	datetime	8	True		(getdate())
	ModifiedDate	datetime	8	True		
	LeadExposure	bit	1	True		

Indexes

Key	Name	Columns	Unique	File Group
	PK_Occupation	OccupationID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateOccupation	True	True	After Update

SQL Script

```

CREATE TABLE [dbo].[Occupation]
(
[OccupationID] [int] NOT NULL IDENTITY(1, 1),
[OccupationName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
[OccupationDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Occupation_CreatedDate] DEFAULT
(getdate()),
[ModifiedDate] [datetime] NULL,
[LeadExposure] [bit] NULL
) ON [UData]
GO
create trigger [dbo].[trUpdateOccupation] on [dbo].[Occupation] AFTER UPDATE
as
begin
if @@rowcount = 0
return
if not update(ModifiedDate) update Occupation set ModifiedDate = getdate() where
OccupationID in (select OccupationID from inserted)

end
GO
ALTER TABLE [dbo].[Occupation] ADD CONSTRAINT [PK_Occupation] PRIMARY KEY CLUSTERED
([OccupationID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'collection of occupation objects',
'SCHEMA', N'dbo', 'TABLE', N'Occupation', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'short description of the occupation',
'SCHEMA', N'dbo', 'TABLE', N'Occupation', 'COLUMN', N'OccupationDescription'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'unique identifier of the occupation',
'SCHEMA', N'dbo', 'TABLE', N'Occupation', 'COLUMN', N'OccupationID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'name of the occupation', 'SCHEMA',
N'dbo', 'TABLE', N'Occupation', 'COLUMN', N'OccupationName'
GO

```

Used By

[dbo].[OccupationNotes]
[dbo].[PersontoOccupation]
[dbo].[usp_InsertOccupation]

[dbo].[OccupationNotes]

MS_Description

table for Occupation notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	0
Created	12:43:48 AM Tuesday, February 17, 2015
Last Modified	12:54:37 AM Tuesday, February 17, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	OccupationNotesID	int	4	False	1 - 1	
 D	OccupationID	int	4	False		
	CreatedDate date the notes where added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	False		

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_OccupationNotes	OccupationNotesID	True	UData

Foreign Keys

Name	Columns
FK_OccupationNotes_Occupation	OccupationID->[dbo].[Occupation].[OccupationID]

SQL Script

```
CREATE TABLE [dbo] . [OccupationNotes]
(
```

Author: liam

```
[OccupationNotesID] [int] NOT NULL IDENTITY(1, 1),
[OccupationID] [int] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_OccupationNotes_CreatedDate] DEFAULT
(getdate()),
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NOT NULL
) ON [UData]
GO
ALTER TABLE [dbo].[OccupationNotes] ADD CONSTRAINT [PK_OccupationNotes] PRIMARY KEY
CLUSTERED ([OccupationNotesID]) ON [UData]
GO
ALTER TABLE [dbo].[OccupationNotes] ADD CONSTRAINT [FK_OccupationNotes_Occupation]
FOREIGN KEY ([OccupationID]) REFERENCES [dbo].[Occupation] ([OccupationID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'table for Occupation notes', 'SCHEMA',
N'dbo', 'TABLE', N'OccupationNotes', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the notes where added', 'SCHEMA',
N'dbo', 'TABLE', N'OccupationNotes', 'COLUMN', N'CreatedDate'
GO
```

Uses

[dbo].[Occupation]

[dbo].[Person]

MS_Description

collection of people and basic attributes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	3446
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	5:31:42 PM Thursday, June 11, 2015

Columns

Key	Name	Data Type	Computed	Max Length (Bytes)	Allow Nulls	Identity	Default
	PersonID unique identifier for each person	int		4	False	1 - 1	
	FirstName Person's first name	varchar(50)		50	False		
	MiddleName Person's middle name or initial	varchar(50)		50	True		
	LastName Person's last name	varchar(50)		50	False		
	BirthDate Person's date of birth	date		3	True		
	Gender Person's gender (i.e. male or female)	char(1)		1	True		
	StatusID	smallint		2	True		
	ForeignTravel 0 = No; 1 = Yes; does the person travel to foreign countries	bit		1	True		
	OutOfSite 0 = No; 1 = Yes; is the person living out of the lead study area.	bit		1	True		
	EatsForeignFood 0 = No; 1 = Yes; does the person eat foreign candy	bit		1	True		

Project> (local)> User databases> LCCHPDev> Tables> dbo.Person

	HistoricChildID ChildID from the access db system	smallint		2	True		
	RetestDate date for the next scheduled test	date		3	True		
	Moved 0 = No; 1 = Yes; has the person moved outside the lead study area	bit		1	True		
	MovedDate Date the person moved outside of the study area	date		3	True		
	isClosed 0 = No; 1 = Yes; is the person's lead study closed	bit		1	True		
	isResolved 0 = No; 1 = Yes; has the lead issue been resolved	bit		1	True		
	GuardianID personID of the person's guardian	int		4	True		
	personCode	smallint		2	True		
	isSmoker 0 = No; 1 = Yes; does the person smoke	bit		1	True		
	CreatedDate Date the record was created	datetime		8	True		(getdate())
	ModifiedDate Date the record was last modified	datetime		8	True		
	Age Calculated age of the person in years	int	True	4	True		
	isClient 0 = No; 1 = Yes; Is the person a participant in the lead study program	bit		1	True		((1))
	NursingMother 0 = No; 1 = Yes; Is the person a nursing mother	bit		1	True		
	Pregnant 0 = No; 1 = Yes; Is the person pregnant	bit		1	True		
	ReleaseStatusID	tinyint		1	True		
	ReviewStatusID	tinyint		1	True		
	EmailAddress Person's email address	varchar(320)		320	True		
	NursingInfant 0 = No; 1 = Yes; Is the person a nursing infant	bit		1	True		
	ClientStatusID	smallint		2	True		

Author: liam

Computed columns

Name	Column definition
Age	([dbo].[udf_CalculateAge]([BirthDate],getdate()))

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_Person	PersonID	True	UData
	NonClusteredIndex-20141220-115023	LastName, Retest-Date		UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdatePerson	True	True	After Update

Check Constraints

Name	On Column	Constraint
ck_Person_BirthDate	BirthDate	([dbo].[udf_DateInThePast]([BirthDate])=(1))
ck_Person_MovedDate	MovedDate	([dbo].[udf_DateInThePast]([MovedDate])=(1) OR [MovedDate] IS NULL)

Foreign Keys

Name	Columns
FK_Person_TargetStatus	ClientStatusID->[dbo].[TargetStatus].[StatusID]
FK_Person_ReviewStatus	ReviewStatusID->[dbo].[ReviewStatus].[ReviewStatusID]

SQL Script

```
CREATE TABLE [dbo].[Person]
(
    [PersonID] [int] NOT NULL IDENTITY(1, 1),
    [FirstName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
    [MiddleName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
    [LastName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
    [BirthDate] [date] NULL,
    [Gender] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
    [StatusID] [smallint] NULL,
    [ForeignTravel] [bit] NULL,
    [OutofSite] [bit] NULL,
```

Project> (local)> User databases> LCCHPDev> Tables> dbo.Person

```
[EatsForeignFood] [bit] NULL,
[HistoricChildID] [smallint] NULL,
[RetestDate] [date] NULL,
[Moved] [bit] NULL,
[MovedDate] [date] NULL,
[isClosed] [bit] NULL,
[isResolved] [bit] NULL,
[GuardianID] [int] NULL,
[personCode] [smallint] NULL,
[isSmoker] [bit] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Person_CreatedDate] DEFAULT (getdate()),
[ModifiedDate] [datetime] NULL,
[Age] AS ([dbo].[udf_CalculateAge](BirthDate),getdate())),
[isClient] [bit] NULL CONSTRAINT [DF_Person_isClient] DEFAULT ((1)),
[NursingMother] [bit] NULL,
[Pregnant] [bit] NULL,
[ReleaseStatusID] [tinyint] NULL,
[ReviewStatusID] [tinyint] NULL,
[EmailAddress] [varchar] (320) COLLATE SQL_Latin1_General_CI_AS NULL,
[NursingInfant] [bit] NULL,
[ClientStatusID] [smallint] NULL
) ON [UData]
GO
create trigger [dbo].[trUpdatePerson] on [dbo].[Person] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update Person set ModifiedDate = getdate() where
PersonID in (select PersonID from inserted)

    end
GO
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [ck_Person_BirthDate] CHECK ((([dbo].[udf_Date-
InThePast](BirthDate))=(1)))
GO
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [ck_Person_MovedDate] CHECK ((([dbo].[udf_Date-
InThePast](MovedDate))=(1) OR [MovedDate] IS NULL))
GO
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [PK_Person] PRIMARY KEY CLUSTERED ([Person-
ID]) ON [UData]
GO
CREATE NONCLUSTERED INDEX [NonClusteredIndex-20141220-115023] ON [dbo].[Person] ([Last-
Name], [RetestDate]) ON [UData]
GO
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [FK_Person_TargetStatus] FOREIGN KEY ([Client-
StatusID]) REFERENCES [dbo].[TargetStatus] ([StatusID])
GO
ALTER TABLE [dbo].[Person] ADD CONSTRAINT [FK_Person_ReviewStatus] FOREIGN KEY ([Review-
StatusID]) REFERENCES [dbo].[ReviewStatus] ([ReviewStatusID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of people and basic
attributes', 'SCHEMA', N'dbo', 'TABLE', N'Person', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'Calculated age of the person in
```

Author: liam

Project> (local)> User databases> LCCHPDev> Tables> dbo.Person

```

years', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'Age'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Person''s date of birth', 'SCHEMA',
N'dbo', 'TABLE', N'Person', 'COLUMN', N'BirthDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Date the record was created',
'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'CreatedDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = No; 1 = Yes; does the person eat
foreign candy', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'EatsForeignFood'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Person''s email address', 'SCHEMA',
N'dbo', 'TABLE', N'Person', 'COLUMN', N'EmailAddress'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Person''s first name', 'SCHEMA',
N'dbo', 'TABLE', N'Person', 'COLUMN', N'FirstName'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = No; 1 = Yes; does the person
travel to foreign countries', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'Foreign-
Travel'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Person''s gender (i.e. male or
female)', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'Gender'
GO
EXEC sp_addextendedproperty N'MS_Description', N'personID of the person''s guardian',
'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'GuardianID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'ChildID from the access db system',
'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'HistoricChildID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = No; 1 = Yes; Is the person a
participant in the lead study program', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN',
N'isClient'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = No; 1 = Yes; is the person's lead
study closed', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'isClosed'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = No; 1 = Yes; has the lead issue
been resolved', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'isResolved'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = No; 1 = Yes; does the person
smoke', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'isSmoker'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Person''s last name', 'SCHEMA',
N'dbo', 'TABLE', N'Person', 'COLUMN', N'LastName'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Person''s middle name or initial',
'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'MiddleName'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Date the record was last modified',
'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'ModifiedDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = No; 1 = Yes; has the person moved
outside the lead study area', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'Moved'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Date the person moved outside of the
study area', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'MovedDate'
GO

```

Author: liam

```
EXEC sp_addextendedproperty N'MS_Description', N'0 = No; 1 = Yes; Is the person a nursing infant', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'NursingInfant'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = No; 1 = Yes; Is the person a nursing mother', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'NursingMother'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = No; 1 = Yes; is the person living out of the lead study area.', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'Outof-Site'
GO
EXEC sp_addextendedproperty N'MS_Description', N'unique identifier for each person', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'PersonID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = No; 1 = Yes; Is the person pregnant', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'Pregnant'
GO
EXEC sp_addextendedproperty N'MS_Description', N'date for the next scheduled test', 'SCHEMA', N'dbo', 'TABLE', N'Person', 'COLUMN', N'RetestDate'
GO
```

Uses

[dbo].[ReviewStatus]
[dbo].[TargetStatus]
[dbo].[udf_CalculateAge]
[dbo].[udf_DateInThePast]

Used By

[dbo].[BloodTestResults]
[dbo].[GiftCard]
[dbo].[PersonHobbyNotes]
[dbo].[PersonNotes]
[dbo].[PersonReleaseNotes]
[dbo].[PersontoAccessAgreement]
[dbo].[PersontoDaycare]
[dbo].[PersontoEmployer]
[dbo].[PersontoEthnicity]
[dbo].[PersontoFamily]
[dbo].[PersontoForeignFood]
[dbo].[PersontoHobby]
[dbo].[PersontoHomeRemedy]
[dbo].[PersontoInsurance]
[dbo].[PersontoLanguage]
[dbo].[PersontoOccupation]
[dbo].[PersontoPerson]
[dbo].[PersontoPhoneNumber]
[dbo].[PersontoProperty]
[dbo].[PersonToTravelCountry]
[dbo].[PersonTravelNotes]
[dbo].[Property]
[dbo].[Questionnaire]
[dbo].[vAdults]

Project> (local)> User databases> LCCHPDev> Tables> dbo.Person

```
[dbo].[vMostRecentBloodTestResults]
[dbo].[vMostRecentQuestionnaires]
[dbo].[vNursingInfants]
[dbo].[vNursingMothers]
[dbo].[vPregnant]
[dbo].[usp_InsertBloodTestResults]
[dbo].[usp_InsertGiftCard]
[dbo].[usp_InsertNewQuestionnaireWebScreen]
[dbo].[usp_InsertPerson]
[dbo].[usp_SLAllBloodTestResultsMetaData]
[dbo].[usp_SICountAdults]
[dbo].[usp_SICountPeopleByAgeGroup]
[dbo].[usp_SLInsertedDataSimplified]
[dbo].[usp_SIStoryReport]
[dbo].[usp_upClientFlag]
[dbo].[usp_upPerson]
```

[dbo].[PersonHobbyNotes]

MS_Description

table for person hobby notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	8
Created	12:58:55 AM Tuesday, April 14, 2015
Last Modified	12:58:55 AM Tuesday, April 14, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	PersonHobbyNotesID	int	4	False	1 - 1	
 D	PersonID	int	4	False		
	CreatedDate date the notes where added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	False		

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_PersonHobbyNotes	PersonHobbyNotesID	True	UData

Foreign Keys

Name	Columns
FK_PersonHobbyNotes_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```
CREATE TABLE [dbo].[PersonHobbyNotes]
(
```

Author: liam

```
[PersonHobbyNotesID] [int] NOT NULL IDENTITY(1, 1),
[PersonID] [int] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PersonHobbyNotes_CreatedDate] DEFAULT
(getdate()),
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NOT NULL
) ON [UData]
GO
ALTER TABLE [dbo].[PersonHobbyNotes] ADD CONSTRAINT [PK_PersonHobbyNotes] PRIMARY KEY
CLUSTERED ([PersonHobbyNotesID]) ON [UData]
GO
ALTER TABLE [dbo].[PersonHobbyNotes] ADD CONSTRAINT [FK_PersonHobbyNotes_Person]
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'table for person hobby notes',
'SCHEMA', N'dbo', 'TABLE', N'PersonHobbyNotes', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the notes where added', 'SCHEMA',
N'dbo', 'TABLE', N'PersonHobbyNotes', 'COLUMN', N'CreatedDate'
GO
```

Uses

[dbo].[Person]

Used By

[dbo].[usp_InsertPersonHobbyNotes]

[dbo].[PersonNotes]

MS_Description

table for person notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	125
Created	10:54:43 AM Saturday, February 14, 2015
Last Modified	3:18:50 PM Sunday, February 15, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	PersonNotesID	int	4	False	1 - 1	
	PersonID	int	4	False		
	CreatedDate date the notes were added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	False		

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersonNotes	PersonNotesID	True	UData

Foreign Keys

Name	Columns
FK_PersonNotes_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```
CREATE TABLE [dbo].[PersonNotes]
(
```

Author: liam

```
[PersonNotesID] [int] NOT NULL IDENTITY(1, 1),
[PersonID] [int] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PersonNotes_CreatedDate] DEFAULT
(getdate()),
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NOT NULL
) ON [UData]
GO
ALTER TABLE [dbo].[PersonNotes] ADD CONSTRAINT [PK_PersonNotes] PRIMARY KEY CLUSTERED
([PersonNotesID]) ON [UData]
GO
ALTER TABLE [dbo].[PersonNotes] ADD CONSTRAINT [FK_PersonNotes_Person] FOREIGN KEY
([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'table for person notes', 'SCHEMA',
N'dbo', 'TABLE', N'PersonNotes', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the notes where added', 'SCHEMA',
N'dbo', 'TABLE', N'PersonNotes', 'COLUMN', N'CreatedDate'
GO
```

Uses

[dbo].[Person]

Used By

[dbo].[usp_InsertPersonNotes]

[dbo].[PersonReleaseNotes]

MS_Description

table for person release notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	37
Created	6:55:41 PM Saturday, April 11, 2015
Last Modified	6:55:41 PM Saturday, April 11, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	PersonReleaseNotesID	int	4	False	1 - 1	
 D	PersonID	int	4	False		
	CreatedDate date the notes where added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	False		

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_PersonReleaseNotes	PersonReleaseNotesID	True	UData

Foreign Keys

Name	Columns
FK_PersonReleaseNotes_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```
CREATE TABLE [dbo].[PersonReleaseNotes]
(
```

Author: liam

```
[PersonReleaseNotesID] [int] NOT NULL IDENTITY(1, 1),
[PersonID] [int] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PersonReleaseNotes_CreatedDate] DEFAULT
(getdate()),
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NOT NULL
) ON [UData]
GO
ALTER TABLE [dbo].[PersonReleaseNotes] ADD CONSTRAINT [PK_PersonReleaseNotes] PRIMARY
KEY CLUSTERED ([PersonReleaseNotesID]) ON [UData]
GO
ALTER TABLE [dbo].[PersonReleaseNotes] ADD CONSTRAINT [FK_PersonReleaseNotes_Person]
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'table for person release notes',
'SCHEMA', N'dbo', 'TABLE', N'PersonReleaseNotes', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the notes where added', 'SCHEMA',
N'dbo', 'TABLE', N'PersonReleaseNotes', 'COLUMN', N'CreatedDate'
GO
```

Uses

[dbo].[Person]

Used By

[dbo].[usp_InsertPersonReleaseNotes]

[dbo].[PersonStatus]

MS_Description

Collection of potential status for person

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	0
Created	1:55:09 PM Saturday, April 11, 2015
Last Modified	1:55:09 PM Saturday, April 11, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	PersonStatusID	tinyint	1	False	1 - 1	
	PersonStatusDescription Detailed description of the person status	varchar(253)	253	True		
	PersonStatusName status for the person	varchar(50)	50	True		
	HistoricPersonStatusID historic status from access database	char(1)	1	True		
	ModifiedDate last modified date for the record	datetime	8	True		
	CreatedDate date the record was created	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersonStatus	PersonStatusID	True	UData

SQL Script

```
CREATE TABLE [dbo].[PersonStatus]
(
[PersonStatusID] [tinyint] NOT NULL IDENTITY(1, 1),
```

Author: liam

Project> (local)> User databases> LCCHPDev> Tables> dbo.PersonStatus

```
[PersonStatusDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[PersonStatusName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[HistoricPersonStatusID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PersonStatus_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[PersonStatus] ADD CONSTRAINT [PK_PersonStatus] PRIMARY KEY CLUSTERED
([PersonStatusID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Collection of potential status for
person', 'SCHEMA', N'dbo', 'TABLE', N'PersonStatus', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the record was created',
'SCHEMA', N'dbo', 'TABLE', N'PersonStatus', 'COLUMN', N'CreatedDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'historic status from access database',
'SCHEMA', N'dbo', 'TABLE', N'PersonStatus', 'COLUMN', N'HistoricPersonStatusID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'last modified date for the record',
'SCHEMA', N'dbo', 'TABLE', N'PersonStatus', 'COLUMN', N'ModifiedDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Detailed description of the person
status', 'SCHEMA', N'dbo', 'TABLE', N'PersonStatus', 'COLUMN', N'PersonStatus-
Description'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'status for the person', 'SCHEMA',
N'dbo', 'TABLE', N'PersonStatus', 'COLUMN', N'PersonStatusName'
GO
```

[dbo].[PersontoAccessAgreement]

MS_Description

linking table for person and access agreement

Properties

Property	Value
File Group	UData
Row Count (~)	1
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	10:56:10 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PersonID	int	4	False	
	AccessAgreementID	int	4	False	
	AccessAgreementDate date the access agreement was signed	date	3	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersontoAccessAgreement	PersonID, Access-AgreementID	True	UData

Foreign Keys

Name	Columns
FK_PersontoAccessAgreement_AccessAgreement	AccessAgreementID->[dbo].[AccessAgreement].[AccessAgreementID]
FK_PersontoAccessAgreement_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```
CREATE TABLE [dbo].[PersontoAccessAgreement]
(
    [PersonID] [int] NOT NULL,
    [AccessAgreementID] [int] NOT NULL,
    [AccessAgreementDate] [date] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_PersontoAccessAgreement_CreatedDate]
        DEFAULT (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[PersontoAccessAgreement] ADD CONSTRAINT [PK_PersontoAccessAgreement]
PRIMARY KEY CLUSTERED ([PersonID], [AccessAgreementID]) ON [UData]
GO
ALTER TABLE [dbo].[PersontoAccessAgreement] ADD CONSTRAINT [FK_PersontoAccessAgreement_AccessAgreement] FOREIGN KEY ([AccessAgreementID]) REFERENCES [dbo].[AccessAgreement] ([AccessAgreementID])
GO
ALTER TABLE [dbo].[PersontoAccessAgreement] ADD CONSTRAINT [FK_PersontoAccessAgreement_Person] FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for person and access agreement', 'SCHEMA', 'dbo', 'TABLE', N'PersontoAccessAgreement', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the access agreement was signed', 'SCHEMA', 'dbo', 'TABLE', N'PersontoAccessAgreement', 'COLUMN', N'AccessAgreementDate'
GO
```

Uses

[dbo].[AccessAgreement]
[dbo].[Person]

Used By

[dbo].[usp_InsertPersontoAccessAgreement]

[dbo].[PersontoDaycare]	

MS_Description

linking table for person and daycare for people attending daycare

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	2
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	10:56:44 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PersonID	int	4	False	
	DaycareID	int	4	False	
	StartDate date the person started attending the daycare	date	3	False	(getdate())
	EndDate date the person stopped attending the daycare	date	3	True	
	DaycareNotes	varchar(3000)	3000	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersontoDaycare	PersonID, DaycareID, StartDate	True	UData

Foreign Keys

Name	Columns

FK_PersontoDaycare_Daycare	DaycareID->[dbo].[Daycare].[DaycareID]
FK_PersontoDaycare_PersontoDaycare	PersonID->[dbo].[Person].[PersonID]

SQL Script

```

CREATE TABLE [dbo].[PersontoDaycare]
(
    [PersonID] [int] NOT NULL,
    [DaycareID] [int] NOT NULL,
    [StartDate] [date] NOT NULL CONSTRAINT [DF_PersontoDaycare_StartDate] DEFAULT
    (getdate()),
    [EndDate] [date] NULL,
    [DaycareNotes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_PersontoDaycare_CreatedDate] DEFAULT
    (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[PersontoDaycare] ADD CONSTRAINT [PK_PersontoDaycare] PRIMARY KEY
CLUSTERED ([PersonID], [DaycareID], [StartDate]) ON [UData]
GO
ALTER TABLE [dbo].[PersontoDaycare] ADD CONSTRAINT [FK_PersontoDaycare_Daycare] FOREIGN
KEY ([DaycareID]) REFERENCES [dbo].[Daycare] ([DaycareID])
GO
ALTER TABLE [dbo].[PersontoDaycare] ADD CONSTRAINT [FK_PersontoDaycare_PersontoDaycare]
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for person and daycare
for people attending daycare', 'SCHEMA', N'dbo', 'TABLE', N'PersontoDaycare', NULL,
NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the person stopped attending the
daycare', 'SCHEMA', N'dbo', 'TABLE', N'PersontoDaycare', 'COLUMN', N'EndDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the person started attending the
daycare', 'SCHEMA', N'dbo', 'TABLE', N'PersontoDaycare', 'COLUMN', N'StartDate'
GO

```

Uses

[dbo].[Daycare]
[dbo].[Person]

Used By

[dbo].[usp_InsertPersontoDaycare]

[dbo].[PersontoEmployer]

MS_Description

linking table for person and employer

Properties

Property	Value
File Group	UData
Row Count (~)	1
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	10:56:48 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PersonID	int	4	False	
	EmployerID	int	4	False	
	StartDate date the person started working for the employer	date	3	False	(getdate())
	EndDate date the person stopped working for the employer	date	3	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersontoEmployer	PersonID, EmployerID, StartDate	True	UData

Foreign Keys

Name	Columns
FK_PersontoEmployer_Employer	EmployerID->[dbo].[Employer].[EmployerID]
FK_PersontoEmployer_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```
CREATE TABLE [dbo].[PersontoEmployer]
(
[PersonID] [int] NOT NULL,
[EmployerID] [int] NOT NULL,
[StartDate] [date] NOT NULL CONSTRAINT [DF_PersontoEmployer_StartDate] DEFAULT
(getdate()),
[EndDate] [date] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PersontoEmployer_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[PersontoEmployer] ADD CONSTRAINT [PK_PersontoEmployer] PRIMARY KEY
CLUSTERED ([PersonID], [EmployerID], [StartDate]) ON [UData]
GO
ALTER TABLE [dbo].[PersontoEmployer] ADD CONSTRAINT [FK_PersontoEmployer_Employer]
FOREIGN KEY ([EmployerID]) REFERENCES [dbo].[Employer] ([EmployerID])
GO
ALTER TABLE [dbo].[PersontoEmployer] ADD CONSTRAINT [FK_PersontoEmployer_Person]
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for person and
employer', 'SCHEMA', N'dbo', 'TABLE', N'PersontoEmployer', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the person stopped working for
the employer', 'SCHEMA', N'dbo', 'TABLE', N'PersontoEmployer', 'COLUMN', N'EndDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the person started working for
the employer', 'SCHEMA', N'dbo', 'TABLE', N'PersontoEmployer', 'COLUMN', N'StartDate'
GO
```

Uses

[dbo].[Employer]
[dbo].[Person]

Used By

[dbo].[usp_InsertPersontoEmployer]

[dbo].[PersontoEthnicity]	

MS_Description

linking table for person and ethnicity

Properties

Property	Value
File Group	UData
Row Count (~)	52
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	2:36:44 PM Saturday, March 28, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PersonID	int	4	False	
	EthnicityID	tinyint	1	False	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersontoEthnicity_1	PersonID, EthnicityID	True	UData

Foreign Keys

Name	Columns
FK_PersontoEthnicity_Ethnicity	EthnicityID->[dbo].[Ethnicity].[EthnicityID]
FK_PersontoEthnicity_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```
CREATE TABLE [dbo].[PersontoEthnicity]
(
[PersonID] [int] NOT NULL,
[EthnicityID] [tinyint] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PersontoEthnicity_CreatedDate] DEFAULT
)
```

Author: liam

```
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[PersontoEthnicity] ADD CONSTRAINT [PK_PersontoEthnicity_1] PRIMARY
KEY CLUSTERED ([PersonID], [EthnicityID]) ON [UData]
GO
ALTER TABLE [dbo].[PersontoEthnicity] ADD CONSTRAINT [FK_PersontoEthnicity_Ethnicity]
FOREIGN KEY ([EthnicityID]) REFERENCES [dbo].[Ethnicity] ([EthnicityID])
GO
ALTER TABLE [dbo].[PersontoEthnicity] ADD CONSTRAINT [FK_PersontoEthnicity_Person]
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for person and
ethnicity', 'SCHEMA', N'dbo', 'TABLE', N'PersontoEthnicity', NULL, NULL
GO
```

Uses

[dbo].[Ethnicity]
[dbo].[Person]

Used By

[dbo].[usp_InsertPersontoEthnicity]
[dbo].[usp_upClientWebScreen]

[dbo].[PersontoFamily]

MS_Description

linking table for person and family tables

Properties

Property	Value
File Group	UData
Row Count (~)	3427
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	10:18:50 PM Sunday, April 19, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PersonID id of the corresponding person	int	4	False	
	FamilyID id of the corresponding family	int	4	False	
	CreatedDate	datetime	8	True	(getdate())
	StartDate	date	3	True	
	EndDate	date	3	True	
	PrimaryContactFamily	bit	1	True	
	ReviewStatusID	tinyint	1	True	
	ModifiedDate	date	3	True	

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersontoFamily	PersonID, FamilyID	True	UData

Foreign Keys

Name	Columns

Author: liam

FK_PersontoFamily_Family	FamilyID->[dbo].[Family].[FamilyID]
FK_PersontoFamily_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```

CREATE TABLE [dbo].[PersontoFamily]
(
    [PersonID] [int] NOT NULL,
    [FamilyID] [int] NOT NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_PersontoFamily_CreatedDate] DEFAULT
    (getdate()),
    [StartDate] [date] NULL,
    [EndDate] [date] NULL,
    [PrimaryContactFamily] [bit] NULL,
    [ReviewStatusID] [tinyint] NULL,
    [ModifiedDate] [date] NULL
) ON [UData]
GO
ALTER TABLE [dbo].[PersontoFamily] ADD CONSTRAINT [PK_PersontoFamily] PRIMARY KEY
CLUSTERED ([PersonID], [FamilyID]) ON [UData]
GO
ALTER TABLE [dbo].[PersontoFamily] ADD CONSTRAINT [FK_PersontoFamily_Family] FOREIGN
KEY ([FamilyID]) REFERENCES [dbo].[Family] ([FamilyID])
GO
ALTER TABLE [dbo].[PersontoFamily] ADD CONSTRAINT [FK_PersontoFamily_Person] FOREIGN
KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for person and family
tables', 'SCHEMA', N'dbo', 'TABLE', N'PersontoFamily', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the corresponding family',
'SCHEMA', N'dbo', 'TABLE', N'PersontoFamily', 'COLUMN', N'FamilyID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the corresponding person',
'SCHEMA', N'dbo', 'TABLE', N'PersontoFamily', 'COLUMN', N'PersonID'
GO

```

Uses

[dbo].[Family]
[dbo].[Person]

Used By

[dbo].[usp_InsertPersontoFamily]
[dbo].[usp_SInsertedDataSimplified]
[dbo].[usp_upClientWebScreen]

[dbo].[PersontoForeignFood]

MS_Description

linking table for person and foreign food (many to many)

Properties

Property	Value
File Group	UData
Row Count (~)	3
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	10:56:48 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PersonID	int	4	False	
	ForeignFoodID	int	4	False	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersontoForeignFood	PersonID, ForeignFoodID	True	UData

Foreign Keys

Name	Columns
FK_PersontoForeignFood_ForeignFood	ForeignFoodID->[dbo].[ForeignFood].[ForeignFoodID]
FK_PersontoForeignFood_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```
CREATE TABLE [dbo].[PersontoForeignFood]
(
    [PersonID] [int] NOT NULL,
    [ForeignFoodID] [int] NOT NULL,
```

```
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PersontoForeignFood_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[PersontoForeignFood] ADD CONSTRAINT [PK_PersontoForeignFood] PRIMARY
KEY CLUSTERED ([PersonID], [ForeignFoodID]) ON [UData]
GO
ALTER TABLE [dbo].[PersontoForeignFood] ADD CONSTRAINT [FK_PersontoForeignFood_Foreign-
Food] FOREIGN KEY ([ForeignFoodID]) REFERENCES [dbo].[ForeignFood] ([ForeignFoodID])
GO
ALTER TABLE [dbo].[PersontoForeignFood] ADD CONSTRAINT [FK_PersontoForeignFood_Person]
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for person and foreign
food (many to many)', 'SCHEMA', N'dbo', 'TABLE', N'PersontoForeignFood', NULL, NULL
GO
```

Uses

[dbo].[ForeignFood]
[dbo].[Person]

Used By

[dbo].[usp_InsertPersontoForeignFood]

[dbo].[PersontoHobby]

MS_Description

linking table for person and hobby

Properties

Property	Value
File Group	UData
Row Count (~)	49
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	10:56:48 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PersonID	int	4	False	
	HobbyID	smallint	2	False	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersontoHobby	PersonID, HobbyID	True	UData

Foreign Keys

Name	Columns
FK_PersontoHobby_Hobby	HobbyID->[dbo].[Hobby].[HobbyID]
FK_PersontoHobby_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```
CREATE TABLE [dbo].[PersontoHobby]
(
    [PersonID] [int] NOT NULL,
    [HobbyID] [smallint] NOT NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_PersontoHobby_CreatedDate] DEFAULT
```

Author: liam

```
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[PersontoHobby] ADD CONSTRAINT [PK_PersontoHobby] PRIMARY KEY
CLUSTERED ([PersonID], [HobbyID]) ON [UData]
GO
ALTER TABLE [dbo].[PersontoHobby] ADD CONSTRAINT [FK_PersontoHobby_Hobby] FOREIGN KEY
([HobbyID]) REFERENCES [dbo].[Hobby] ([HobbyID])
GO
ALTER TABLE [dbo].[PersontoHobby] ADD CONSTRAINT [FK_PersontoHobby_Person] FOREIGN KEY
([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for person and hobby',
'SCHEMA', N'dbo', 'TABLE', N'PersontoHobby', NULL, NULL
GO
```

Uses

[dbo].[Hobby]
[dbo].[Person]

Used By

[dbo].[usp_InsertPersontoHobby]
[dbo].[usp_upClientWebScreen]

[dbo].[PersontoHomeRemedy]

MS_Description

linking table for perosn and home remedy

Properties

Property	Value
File Group	UData
Row Count (~)	2
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	10:56:48 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PersonID	int	4	False	
	HomeRemedyID	int	4	False	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersontoHomeRemedy	PersonID, Home-RemedyID	True	UData

Foreign Keys

Name	Columns
FK_PersontoHomeRemedy_PersontoHomeRemedy	HomeRemedyID->[dbo].[HomeRemedy].[HomeRemedy-ID]
FK_PersontoHomeRemedy_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```
CREATE TABLE [dbo].[PersontoHomeRemedy]
(
    [PersonID] [int] NOT NULL,
```

Author: liam

```
[HomeRemedyID] [int] NOT NULL,  
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PersontoHomeRemedy_CreatedDate] DEFAULT  
(getdate())  
) ON [UData]  
GO  
ALTER TABLE [dbo].[PersontoHomeRemedy] ADD CONSTRAINT [PK_PersontoHomeRemedy] PRIMARY  
KEY CLUSTERED ([PersonID], [HomeRemedyID]) ON [UData]  
GO  
ALTER TABLE [dbo].[PersontoHomeRemedy] ADD CONSTRAINT [FK_PersontoHomeRemedy_Personto-  
HomeRemedy] FOREIGN KEY ([HomeRemedyID]) REFERENCES [dbo].[HomeRemedy] ([HomeRemedyID])  
GO  
ALTER TABLE [dbo].[PersontoHomeRemedy] ADD CONSTRAINT [FK_PersontoHomeRemedy_Person]  
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])  
GO  
EXEC sp_addextendedproperty N'MS_Description', N'linking table for person and home  
remedy', 'SCHEMA', 'dbo', 'TABLE', N'PersontoHomeRemedy', NULL, NULL  
GO
```

Uses

[dbo].[HomeRemedy]
[dbo].[Person]

Used By

[dbo].[usp_InsertPersontoHomeRemedy]

[dbo].[PersontoInsurance]

MS_Description

linking table for person and insurance

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	1
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	10:56:48 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PersonID	int	4	False	
	InsuranceID	smallint	2	False	
	StartDate Date the person started the insurance policy with the provider	date	3	True	
	EndDate Date the person stopped the insurance policy with the provider	date	3	True	
	GroupID insurance company and policy group id	varchar(20)	20	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersontoInsurance	PersonID, InsuranceID	True	UData

Foreign Keys

Name	Columns
FK_PersontoInsurance_PersontoInsurance	InsuranceID->[dbo].[InsuranceProvider].[InsuranceProviderID]

FK_PersontoInsurance_Person	PersonID->[dbo].[Person].[PersonID]
-----------------------------	-------------------------------------

SQL Script

```

CREATE TABLE [dbo].[PersontoInsurance]
(
    [PersonID] [int] NOT NULL,
    [InsuranceID] [smallint] NOT NULL,
    [StartDate] [date] NULL,
    [EndDate] [date] NULL,
    [GroupID] [varchar] (20) COLLATE SQL_Latin1_General_CI_AS NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_PersontoInsurance_CreatedDate] DEFAULT
    (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[PersontoInsurance] ADD CONSTRAINT [PK_PersontoInsurance] PRIMARY KEY
CLUSTERED ([PersonID], [InsuranceID]) ON [UData]
GO
ALTER TABLE [dbo].[PersontoInsurance] ADD CONSTRAINT [FK_PersontoInsurance_Personto-
Insurance] FOREIGN KEY ([InsuranceID]) REFERENCES [dbo].[InsuranceProvider] ([Insurance-
ProviderID])
GO
ALTER TABLE [dbo].[PersontoInsurance] ADD CONSTRAINT [FK_PersontoInsurance_Person]
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_adddextendedproperty N'MS_Description', N'linking table for person and
insurance', 'SCHEMA', N'dbo', 'TABLE', N'PersontoInsurance', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Date the person stopped the insurance
policy with the provider', 'SCHEMA', N'dbo', 'TABLE', N'PersontoInsurance', 'COLUMN',
N'EndDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'insurance company and policy group
id', 'SCHEMA', N'dbo', 'TABLE', N'PersontoInsurance', 'COLUMN', N'GroupID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Date the person started the insurance
policy with the provider', 'SCHEMA', N'dbo', 'TABLE', N'PersontoInsurance', 'COLUMN',
N'StartDate'
GO

```

Uses

[dbo].[InsuranceProvider]
 [dbo].[Person]

Used By

[dbo].[usp_InsertPersontoInsurance]

[dbo].[PersontoLanguage]

MS_Description

linking table for person and language

Properties

Property	Value
File Group	UData
Row Count (~)	203
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	10:56:48 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PersonID	int	4	False	
	LanguageID	tinyint	1	False	
	isPrimaryLanguage 0 = no; 1 = yes; is this language the person's primary language	bit	1	False	((1))
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersontoLanguage	PersonID, LanguageID	True	UData

Foreign Keys

Name	Columns
FK_PersontoLanguage_Language	LanguageID->[dbo].[Language].[LanguageID]
FK_PersontoLanguage_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```
CREATE TABLE [dbo].[PersontoLanguage]
(
[PersonID] [int] NOT NULL,
[LanguageID] [tinyint] NOT NULL,
[isPrimaryLanguage] [bit] NOT NULL CONSTRAINT [DF_PersontoLanguage_isPrimaryLanguage]
DEFAULT ((1)),
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PersontoLanguage_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[PersontoLanguage] ADD CONSTRAINT [PK_PersontoLanguage] PRIMARY KEY
CLUSTERED ([PersonID], [LanguageID]) ON [UData]
GO
ALTER TABLE [dbo].[PersontoLanguage] ADD CONSTRAINT [FK_PersontoLanguage_Language]
FOREIGN KEY ([LanguageID]) REFERENCES [dbo].[Language] ([LanguageID])
GO
ALTER TABLE [dbo].[PersontoLanguage] ADD CONSTRAINT [FK_PersontoLanguage_Person]
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for person and
language', 'SCHEMA', N'dbo', 'TABLE', N'PersontoLanguage', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes; is this language the
person''s primary language', 'SCHEMA', N'dbo', 'TABLE', N'PersontoLanguage', 'COLUMN',
N'isPrimaryLanguage'
GO
```

Uses

[dbo].[Language]
[dbo].[Person]

Used By

[dbo].[usp_InsertPersontoLanguage]
[dbo].[usp_SIEditClientInfoWebScreenInformation]

[dbo].[PersontoOccupation]

MS_Description

linking table for person and occupatoin

Properties

Property	Value
File Group	UData
Row Count (~)	7
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	12:53:47 AM Tuesday, February 17, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PersonID	int	4	False	
	OccupationID	int	4	False	
	StartDate date the person started the occupation	date	3	False	(getdate())
	EndDate date the person ceased the occupation	date	3	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersontoOccupation	PersonID, Occupation-ID, StartDate	True	UData

Foreign Keys

Name	Columns
FK_PersontoOccupation_Occupation	OccupationID->[dbo].[Occupation].[OccupationID]
FK_PersontoOccupation_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```
CREATE TABLE [dbo].[PersontoOccupation]
(
[PersonID] [int] NOT NULL,
[OccupationID] [int] NOT NULL,
[StartDate] [date] NOT NULL CONSTRAINT [DF_PersontoOccupation_StartDate] DEFAULT
(getdate()),
[EndDate] [date] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PersontoOccupation_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[PersontoOccupation] ADD CONSTRAINT [PK_PersontoOccupation] PRIMARY
KEY CLUSTERED ([PersonID], [OccupationID], [StartDate]) ON [UData]
GO
ALTER TABLE [dbo].[PersontoOccupation] ADD CONSTRAINT [FK_PersontoOccupation_-_
Occupation] FOREIGN KEY ([OccupationID]) REFERENCES [dbo].[Occupation] ([OccupationID])
GO
ALTER TABLE [dbo].[PersontoOccupation] ADD CONSTRAINT [FK_PersontoOccupation_Person]
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for person and
occupatioin', 'SCHEMA', N'dbo', 'TABLE', N'PersontoOccupation', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the person ceased the
occupation', 'SCHEMA', N'dbo', 'TABLE', N'PersontoOccupation', 'COLUMN', N'EndDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the person started the
occupation', 'SCHEMA', N'dbo', 'TABLE', N'PersontoOccupation', 'COLUMN', N'StartDate'
GO
```

Uses

[dbo].[Occupation]
[dbo].[Person]

Used By

[dbo].[usp_InsertPersontoOccupation]
[dbo].[usp_upClientWebScreen]
[dbo].[usp_upOccupation]

[dbo].[PersontoPerson]

MS_Description

collection of relationships between people

Properties

Property	Value
File Group	UData
Row Count (~)	2
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	1:27:15 AM Thursday, March 26, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	Person1ID	int	4	False	
	Person2ID	int	4	False	
	RelationshipTypeID relationshipType is how P1 relates to P2	int	4	False	
	isGuardian isGuardian is 1 if P1 is P2's guardian	bit	1	True	
	isPrimaryContact isPrimaryContact is 1 if P1 is P2's primary-Contact	bit	1	True	
	CreatedDate	datetime	8	True	(getdate())
	ModifiedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersontoPerson	Person1ID, Person2ID, RelationshipTypeID	True	UData

Foreign Keys

Name	Columns

Author: liam

FK_PersontoPerson_Person1ID 1st person in the relationship	Person1ID->[dbo].[Person].[PersonID]
FK_PersontoPerson_Person2ID 2nd person in the relationship	Person2ID->[dbo].[Person].[PersonID]
FK_PersontoPerson_RelationshipTypeID how is person1 related to person2	RelationshipTypeID->[dbo].[RelationshipType].[RelationshipTypeID]

SQL Script

```

CREATE TABLE [dbo].[PersontoPerson]
(
    [Person1ID] [int] NOT NULL,
    [Person2ID] [int] NOT NULL,
    [RelationshipTypeID] [int] NOT NULL,
    [isGuardian] [bit] NULL,
    [isPrimaryContact] [bit] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_PersontoPerson_CreatedDate] DEFAULT
    (getdate()),
    [ModifiedDate] [datetime] NULL CONSTRAINT [DF_PersontoPerson_ModifiedDate] DEFAULT
    (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[PersontoPerson] ADD CONSTRAINT [PK_PersontoPerson] PRIMARY KEY
CLUSTERED ([Person1ID], [Person2ID], [RelationshipTypeID]) ON [UData]
GO
ALTER TABLE [dbo].[PersontoPerson] ADD CONSTRAINT [FK_PersontoPerson_Person1ID] FOREIGN
KEY ([Person1ID]) REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[PersontoPerson] ADD CONSTRAINT [FK_PersontoPerson_Person2ID] FOREIGN
KEY ([Person2ID]) REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[PersontoPerson] ADD CONSTRAINT [FK_PersontoPerson_RelationshipType]
FOREIGN KEY ([RelationshipTypeID]) REFERENCES [dbo].[RelationshipType] ([Relationship-
TypeID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of relationships between
people', 'SCHEMA', N'dbo', 'TABLE', N'PersontoPerson', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'isGuardian is 1 if P1 is P2''s
guardian', 'SCHEMA', N'dbo', 'TABLE', N'PersontoPerson', 'COLUMN', N'isGuardian'
GO
EXEC sp_addextendedproperty N'MS_Description', N'isPrimaryContact is 1 if P1 is P2''s
primaryContact', 'SCHEMA', N'dbo', 'TABLE', N'PersontoPerson', 'COLUMN', N'isPrimary-
Contact'
GO
EXEC sp_addextendedproperty N'MS_Description', N'relationshipType is how P1 relates to
P2', 'SCHEMA', N'dbo', 'TABLE', N'PersontoPerson', 'COLUMN', N'RelationshipTypeID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'1st person in the relationship',
'SCHEMA', N'dbo', 'TABLE', N'PersontoPerson', 'CONSTRAINT', N'FK_PersontoPerson_-
Person1ID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'2nd person in the relationship',
'SCHEMA', N'dbo', 'TABLE', N'PersontoPerson', 'CONSTRAINT', N'FK_PersontoPerson_-
Person2ID'
GO

```

```
EXEC sp_addextendedproperty N'MS_Description', N'how is person1 related to person2',
'SCHEMA', N'dbo', 'TABLE', N'PersontoPerson', 'CONSTRAINT', N'FK_PersontoPerson_-
RelationshipType'
```

```
GO
```

Uses

[dbo].[Person]
[dbo].[RelationshipType]

Used By

[dbo].[usp_InsertPersontoPerson]

[dbo].[PersontoPhoneNumber]

MS_Description

linking table for person and phonenumbers

Properties

Property	Value
File Group	UData
Row Count (~)	4
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	7:07:13 PM Saturday, April 4, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PersonID	int	4	False	
	PhoneNumberID	int	4	False	
	NumberPriority order which this number should be used to contact the person (1 being first, 2 being 2nd ...)	tinyint	1	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersontoPhoneNumber	PersonID, Phone-NumberID	True	UData

Foreign Keys

Name	Columns
FK_PersontoPhoneNumber_Person	PersonID->[dbo].[Person].[PersonID]
FK_PersontoPhoneNumber_PhoneNumber	PhoneNumberID->[dbo].[PhoneNumber].[PhoneNumberID]

SQL Script

```
CREATE TABLE [dbo].[PersontoPhoneNumber]
(
    [PersonID] [int] NOT NULL,
    [PhoneNumberID] [int] NOT NULL,
    [NumberPriority] [tinyint] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_PersontoPhoneNumber_CreatedDate] DEFAULT
    (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[PersontoPhoneNumber] ADD CONSTRAINT [PK_PersontoPhoneNumber] PRIMARY
KEY CLUSTERED ([PersonID], [PhoneNumberID]) ON [UData]
GO
ALTER TABLE [dbo].[PersontoPhoneNumber] ADD CONSTRAINT [FK_PersontoPhoneNumber_Person]
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[PersontoPhoneNumber] ADD CONSTRAINT [FK_PersontoPhoneNumber_Phone-
Number] FOREIGN KEY ([PhoneNumberID]) REFERENCES [dbo].[PhoneNumber] ([PhoneNumberID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for person and
phonenumbers', 'SCHEMA', N'dbo', 'TABLE', N'PersontoPhoneNumber', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'order which this number should be used
to contact the person (1 being first, 2 being 2nd . . . )', 'SCHEMA', N'dbo', 'TABLE',
N'PersontoPhoneNumber', 'COLUMN', N'NumberPriority'
GO
```

Uses

[dbo].[Person]
[dbo].[PhoneNumber]

Used By

[dbo].[usp_InsertPersontoPhoneNumber]

[dbo].[PersontoProperty]	

MS_Description

linking table for person and property - indicating when a person occupied a property

Properties

Property	Value
File Group	UData
Row Count (~)	4765
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	10:56:57 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	PersonID	int	4	False		
	PropertyID	int	4	False		
	StartDate date the person started occupying the property	date	3	False		(getdate())
	EndDate date the person stopped occupying the property	date	3	True		
	isPrimaryResidence	bit	1	True		
	FamilyID Primary family id mainly from legacy system	int	4	True		
	PersontoPropertyID	int	4	False	1 - 1	
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersontoProperty	PersontoPropertyID	True	UData

Foreign Keys

Name	Columns
FK_PersontoProperty_Person	PersonID->[dbo].[Person].[PersonID]
FK_PersontoProperty_Property	PropertyID->[dbo].[Property].[PropertyID]

SQL Script

```

CREATE TABLE [dbo].[PersontoProperty]
(
    [PersonID] [int] NOT NULL,
    [PropertyID] [int] NOT NULL,
    [StartDate] [date] NOT NULL CONSTRAINT [DF_PersontoProperty_StartDate] DEFAULT
    (getdate()),
    [EndDate] [date] NULL,
    [isPrimaryResidence] [bit] NULL,
    [FamilyID] [int] NULL,
    [PersontoPropertyID] [int] NOT NULL IDENTITY(1, 1),
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_PersontoProperty_CreatedDate] DEFAULT
    (getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[PersontoProperty] ADD CONSTRAINT [PK_PersontoProperty] PRIMARY KEY
CLUSTERED ([PersontoPropertyID]) ON [UData]
GO
ALTER TABLE [dbo].[PersontoProperty] ADD CONSTRAINT [FK_PersontoProperty_Person]
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[PersontoProperty] ADD CONSTRAINT [FK_PersontoProperty_Property]
FOREIGN KEY ([PropertyID]) REFERENCES [dbo].[Property] ([PropertyID])
GO
EXEC sp_adddextendedproperty N'MS_Description', N'linking table for person and property
- indicating when a person occupied a property', 'SCHEMA', N'dbo', 'TABLE', N'Personto-
Property', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the person stopped occupying the
property', 'SCHEMA', N'dbo', 'TABLE', N'PersontoProperty', 'COLUMN', N'EndDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Primary family id mainly from legacy
system', 'SCHEMA', N'dbo', 'TABLE', N'PersontoProperty', 'COLUMN', N'FamilyID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the person started occupying the
property', 'SCHEMA', N'dbo', 'TABLE', N'PersontoProperty', 'COLUMN', N'StartDate'
GO

```

Uses

[dbo].[Person]
[dbo].[Property]

Used By

[dbo].[usp_InsertPersontoProperty]

[dbo].[PersonToTravelCountry]

MS_Description

linking table for person and country traveled too

Properties

Property	Value
File Group	UData
Row Count (~)	2
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	10:56:57 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PersonID	int	4	False	
	CountryID	tinyint	1	False	
	StartDate date the person entered the country	date	3	False	(getdate())
	EndDate date the person left the country	date	3	True	
	CreatedDate	datetime	8	True	(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_PersonToTravelCountry	PersonID, CountryID, StartDate	True	UData

Foreign Keys

Name	Columns
FK_PersonToTravelCountry_Country	CountryID->[dbo].[Country].[CountryID]
FK_PersonToTravelCountry_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```
CREATE TABLE [dbo].[PersonToTravelCountry]
(
[PersonID] [int] NOT NULL,
[CountryID] [tinyint] NOT NULL,
[StartDate] [date] NOT NULL CONSTRAINT [DF_PersonToTravelCountry_StartDate] DEFAULT
(getdate()),
[EndDate] [date] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PersonToTravelCountry_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[PersonToTravelCountry] ADD CONSTRAINT [PK_PersonToTravelCountry]
PRIMARY KEY CLUSTERED ([PersonID], [CountryID], [StartDate]) ON [UData]
GO
ALTER TABLE [dbo].[PersonToTravelCountry] ADD CONSTRAINT [FK_PersonToTravelCountry_Country]
FOREIGN KEY ([CountryID]) REFERENCES [dbo].[Country] ([CountryID])
GO
ALTER TABLE [dbo].[PersonToTravelCountry] ADD CONSTRAINT [FK_PersonToTravelCountry_Person]
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for person and country
traveled too', 'SCHEMA', N'dbo', 'TABLE', N'PersonToTravelCountry', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the person left the country',
'SCHEMA', N'dbo', 'TABLE', N'PersonToTravelCountry', 'COLUMN', N'EndDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the person entered the country',
'SCHEMA', N'dbo', 'TABLE', N'PersonToTravelCountry', 'COLUMN', N'StartDate'
GO
```

Uses

[dbo].[Country]
[dbo].[Person]

Used By

[dbo].[usp_InsertPersonToTravelCountry]

[dbo].[PersonTravelNotes]

MS_Description

table for person Travel notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	36
Created	7:08:32 PM Saturday, April 11, 2015
Last Modified	7:08:32 PM Saturday, April 11, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	PersonTravelNotesID	int	4	False	1 - 1	
 D	PersonID	int	4	False		
	CreatedDate date the notes where added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	False		

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_PersonTravelNotes	PersonTravelNotesID	True	UData

Foreign Keys

Name	Columns
FK_PersonTravelNotes_Person	PersonID->[dbo].[Person].[PersonID]

SQL Script

```
CREATE TABLE [dbo].[PersonTravelNotes]
(

```

Author: liam

```
[PersonTravelNotesID] [int] NOT NULL IDENTITY(1, 1),
[PersonID] [int] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PersonTravelNotes_CreatedDate] DEFAULT
(getdate()),
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NOT NULL
) ON [UData]
GO
ALTER TABLE [dbo].[PersonTravelNotes] ADD CONSTRAINT [PK_PersonTravelNotes] PRIMARY KEY
CLUSTERED ([PersonTravelNotesID]) ON [UData]
GO
ALTER TABLE [dbo].[PersonTravelNotes] ADD CONSTRAINT [FK_PersonTravelNotes_Person]
FOREIGN KEY ([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'table for person Travel notes',
'SCHEMA', N'dbo', 'TABLE', N'PersonTravelNotes', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the notes where added', 'SCHEMA',
N'dbo', 'TABLE', N'PersonTravelNotes', 'COLUMN', N'CreatedDate'
GO
```

Uses

[dbo].[Person]

Used By

[dbo].[usp_InsertPersonTravelNotes]

[dbo].[PhoneNumber]

MS_Description

collection of phone number objects

Properties

Property	Value
File Group	UData
Row Count (~)	73
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	7:07:13 PM Saturday, April 4, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	PhoneNumberID	int	4	False	1 - 1	
	CountryCode code for the country	tinyint	1	False		((1))
	PhoneNumber telephone number	bigint	8	False		
	PhoneNumberTypeID	tinyint	1	True		
	CreatedDate	datetime	8	True		(getdate())
	ModifiedDate	datetime	8	True		

Indexes

Key	Name	Columns	Unique	File Group
	PK_Phonenumber	PhoneNumberID	True	UData
	IX_Phonenumber	PhoneNumber	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdatePhoneNumber	True	True	After Update

Foreign Keys

Name	Columns
FK_PhoneNumber_PhoneNumber	PhoneNumberTypeID->[dbo].[PhoneNumberType].[PhoneNumberTypeID]

SQL Script

```

CREATE TABLE [dbo].[PhoneNumber]
(
    [PhoneNumberID] [int] NOT NULL IDENTITY(1, 1),
    [CountryCode] [tinyint] NOT NULL CONSTRAINT [DF_PhoneNumber_CountryCode] DEFAULT ((1)),
    [PhoneNumber] [bigint] NOT NULL,
    [PhoneNumberTypeID] [tinyint] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_PhoneNumber_CreatedDate] DEFAULT
    (getdate()),
    [ModifiedDate] [datetime] NULL
) ON [UData]
GO
create trigger [dbo].[trUpdatePhoneNumber] on [dbo].[PhoneNumber] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update PhoneNumber set ModifiedDate = getdate() where
    PhoneNumberID in (select PhoneNumberID from inserted)

    end
GO
ALTER TABLE [dbo].[PhoneNumber] ADD CONSTRAINT [PK_PhoneNumber] PRIMARY KEY CLUSTERED
([PhoneNumberID]) ON [UData]
GO
ALTER TABLE [dbo].[PhoneNumber] ADD CONSTRAINT [IX_PhoneNumber] UNIQUE NONCLUSTERED
([PhoneNumber]) ON [UData]
GO
ALTER TABLE [dbo].[PhoneNumber] ADD CONSTRAINT [FK_PhoneNumber_PhoneNumber] FOREIGN KEY
([PhoneNumberTypeID]) REFERENCES [dbo].[PhoneNumberType] ([PhoneNumberTypeID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of phone number objects',
'SCHEMA', N'dbo', 'TABLE', N'PhoneNumber', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'code for the country', 'SCHEMA',
N'dbo', 'TABLE', N'PhoneNumber', 'COLUMN', N'CountryCode'
GO
EXEC sp_addextendedproperty N'MS_Description', N'telephone number', 'SCHEMA', N'dbo',
'TABLE', N'PhoneNumber', 'COLUMN', N'PhoneNumber'
GO

```

Uses

[dbo].[PhoneNumberType]

Author: liam

Used By

[dbo].[FamilytoPhoneNumber]
[dbo].[PersontoPhoneNumber]
[dbo].[usp_InsertPhoneNumber]
[dbo].[udf_SIFamilyPhoneNumber]

[dbo].[PhoneNumberType]	

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	4
Created	11:30:26 PM Friday, December 19, 2014
Last Modified	9:34:40 AM Saturday, April 18, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	PhoneNumberTypeID	tinyint	1	False	1 - 1	
	PhoneNumberTypeName	varchar(50)	50	True		
	CreatedDate	datetime	8	True		(getdate())
	ModifiedDate	datetime	8	True		
	HistoricPhoneCode	char(1)	1	True		

Indexes

Key	Name	Columns	Unique	File Group
	PK_PhoneNumberType	PhoneNumberTypeID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdatePhoneNumberType	True	True	After Update

SQL Script

```

CREATE TABLE [dbo].[PhoneNumberType]
(
    [PhoneNumberTypeID] [tinyint] NOT NULL IDENTITY(1, 1),
    [PhoneNumberTypeName] [varchar](50) COLLATE SQL_Latin1_General_CI_AS NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_PhoneNumberType_CreatedDate] DEFAULT (getdate()),
    [ModifiedDate] [datetime] NULL,
    [HistoricPhoneCode] [char](1) COLLATE SQL_Latin1_General_CI_AS NULL
)

```

Author: liam

```
) ON [UData]
GO
create trigger [dbo].[trUpdatePhoneNumberType] on [dbo].[PhoneNumberType] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update PhoneNumberType set ModifiedDate = getdate()
where PhoneNumberTypeID in (select PhoneNumberTypeID from inserted)

    end
GO
ALTER TABLE [dbo].[PhoneNumberType] ADD CONSTRAINT [PK_PhoneNumberType] PRIMARY KEY
CLUSTERED ([PhoneNumberTypeID]) ON [UData]
GO
```

Used By

[dbo].[PhoneNumber]
[dbo].[usp_InsertNewFamilyWebScreen]
[dbo].[usp_InsertPhoneNumberType]
[dbo].[usp_upFamilyWebScreen]

 [dbo].[Property]	

MS_Description

collection of properties and basic attributes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	10916
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	10:57:42 PM Monday, June 1, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	PropertyID unique identifier for the property object	int	4	False	1 - 1	
	ConstructionTypeID	tinyint	1	True		
	AreaID	int	4	True		
	isinHistoricDistrict	bit	1	True		
	isRemodeled	bit	1	True		
	RemodelDate	date	3	True		
	isinCityLimits	bit	1	True		
	StreetNumber	varchar(15)	15	True		
	AddressLine1	varchar(100)	100	True		
	StreetSuffix	varchar(20)	20	True		
	AddressLine2	varchar(100)	100	True		
	City	varchar(50)	50	True		
	State	char(2)	2	True		

	Zipcode	varchar(12)	12	True		
 	OwnerID	int	4	True		
	isOwnerOccupied	bit	1	True		
	ReplacedPipesFaucets	tinyint	1	True		
	TotalRemediationCosts	money	8	True		
	isResidential	bit	1	True		
	isCurrentlyBeingRemodeled	bit	1	True		
	hasPeelingChippingPaint	bit	1	True		
	County	varchar(50)	50	True		
	isRental	bit	1	True		
	HistoricPropertyID	smallint	2	True		
	CreatedDate	datetime	8	True		(getdate())
	ModifiedDate	datetime	8	True		
	YearBuilt	date	3	True		
	Street	varchar(50)	50	True		
 	ReviewStatusID	tinyint	1	True		
	AssessorsOfficeID Identification number from Assessor's office	varchar(50)	50	True		
	KidsFirstID Kids First identification number	int	4	True		
 	CleanUPStatusID Cleanup status id for the current property cleanup state	tinyint	1	True		
	OwnerContactInformation basic contact information for the property owner	varchar(1000)	1000	True		

Indexes

Key	Name	Columns	Unique	File Group
 	PK_Property	PropertyID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateProperty	True	True	After Update

Foreign Keys

Name	No Check	Columns
FK_Property_Area		AreaID->[dbo].[Area].[AreaID]
FK_Property_CleanupStatus	True	CleanUPStatusID->[dbo].[CleanupStatus].[CleanupStatusID]
FK_Property_ConstructionType		ConstructionTypeID->[dbo].[ConstructionType].[ConstructionTypeID]
FK_Property_Person		OwnerID->[dbo].[Person].[PersonID]
FK_Property_ReleaseStatus		ReviewStatusID->[dbo].[ReleaseStatus].[ReleaseStatusID]

SQL Script

```

CREATE TABLE [dbo].[Property]
(
[PropertyID] [int] NOT NULL IDENTITY(1, 1),
[ConstructionTypeID] [tinyint] NULL,
[AreaID] [int] NULL,
[isinHistoricDistrict] [bit] NULL,
[isRemodeled] [bit] NULL,
[RemodelDate] [date] NULL,
[isinCityLimits] [bit] NULL,
[StreetNumber] [varchar] (15) COLLATE SQL_Latin1_General_CI_AS NULL,
[AddressLine1] [varchar] (100) COLLATE SQL_Latin1_General_CI_AS NULL,
[StreetSuffix] [varchar] (20) COLLATE SQL_Latin1_General_CI_AS NULL,
[AddressLine2] [varchar] (100) COLLATE SQL_Latin1_General_CI_AS NULL,
[City] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[State] [char] (2) COLLATE SQL_Latin1_General_CI_AS NULL,
[Zipcode] [varchar] (12) COLLATE SQL_Latin1_General_CI_AS NULL,
[OwnerID] [int] NULL,
[isOwnerOccupied] [bit] NULL,
[ReplacedPipesFaucets] [tinyint] NULL,
[TotalRemediationCosts] [money] NULL,
[isResidential] [bit] NULL,
[isCurrentlyBeingRemodeled] [bit] NULL,
[hasPeelingChippingPaint] [bit] NULL,
[County] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[isRental] [bit] NULL,
[HistoricPropertyID] [smallint] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Property_CreatedDate] DEFAULT (getdate()),
[ModifiedDate] [datetime] NULL,
[YearBuilt] [date] NULL,
[Street] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[ReviewStatusID] [tinyint] NULL,
[AssessorsOfficeID] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,

```

Project> (local)> User databases> LCCHPDev> Tables> dbo.Property

```
[KidsFirstID] [int] NULL,
[CleanUPStatusID] [tinyint] NULL,
[OwnerContactInformation] [varchar] (1000) COLLATE SQL_Latin1_General_CI_AS NULL
) ON [UData]
GO
create trigger [dbo].[trUpdateProperty] on [dbo].[Property] AFTER UPDATE
as
begin
if @@rowcount = 0
return
if not update(ModifiedDate) update Property set ModifiedDate = getdate() where
PropertyID in (select PropertyID from inserted)

end
GO
ALTER TABLE [dbo].[Property] ADD CONSTRAINT [PK_Property] PRIMARY KEY CLUSTERED
([PropertyID]) ON [UData]
GO
ALTER TABLE [dbo].[Property] ADD CONSTRAINT [FK_Property_Area] FOREIGN KEY ([AreaID])
REFERENCES [dbo].[Area] ([AreaID])
GO
ALTER TABLE [dbo].[Property] WITH NOCHECK ADD CONSTRAINT [FK_Property_CleanupStatus]
FOREIGN KEY ([CleanUPStatusID]) REFERENCES [dbo].[CleanupStatus] ([CleanUpStatusID])
GO
ALTER TABLE [dbo].[Property] ADD CONSTRAINT [FK_Property_ConstructionType] FOREIGN KEY
([ConstructionTypeID]) REFERENCES [dbo].[ConstructionType] ([ConstructionTypeID])
GO
ALTER TABLE [dbo].[Property] ADD CONSTRAINT [FK_Property_Person] FOREIGN KEY ([Owner-
ID]) REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Property] ADD CONSTRAINT [FK_Property_ReleaseStatus] FOREIGN KEY
([ReviewStatusID]) REFERENCES [dbo].[ReleaseStatus] ([ReleaseStatusID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of properties and basic
attributes', 'SCHEMA', N'dbo', 'TABLE', N'Property', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'Identification number from Assessor''s
office', 'SCHEMA', N'dbo', 'TABLE', N'Property', 'COLUMN', N'AssessorsOfficeID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Cleanup status id for the current
property cleanup state', 'SCHEMA', N'dbo', 'TABLE', N'Property', 'COLUMN', N'Clean-
UPStatusID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Kids First identification number',
'SCHEMA', N'dbo', 'TABLE', N'Property', 'COLUMN', N'KidsFirstID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'basic contact information for the
property owner', 'SCHEMA', N'dbo', 'TABLE', N'Property', 'COLUMN', N'OwnerContact-
Information'
GO
EXEC sp_addextendedproperty N'MS_Description', N'unique identifier for the property
object', 'SCHEMA', N'dbo', 'TABLE', N'Property', 'COLUMN', N'PropertyID'
GO
```

Uses

[dbo].[Area]
[dbo].[CleanupStatus]
[dbo].[ConstructionType]
[dbo].[Person]
[dbo].[ReleaseStatus]

Used By

[dbo].[AccessAgreement]
[dbo].[ContractortoProperty]
[dbo].[DaycaretoProperty]
[dbo].[EmployertoProperty]
[dbo].[EnvironmentalInvestigation]
[dbo].[FamilytoProperty]
[dbo].[PersontoProperty]
[dbo].[PropertyNotes]
[dbo].[PropertySampleResults]
[dbo].[PropertystoCleanupStatus]
[dbo].[PropertystoHouseholdSourcesofLead]
[dbo].[PropertytoMedium]
[dbo].[Remediation]
[dbo].[usp_InsertProperty]
[dbo].[usp_upProperty]
[dbo].[udf_DoesPropertyExist]

[dbo].[PropertyLinkType]

MS_Description

Collection of property link types

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	0
Created	3:33:40 PM Saturday, April 11, 2015
Last Modified	5:59:48 PM Saturday, April 11, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	PropertyLinkTypeID	tinyint	1	False	1 - 1	
	PropertyLinkTypeDescription Detailed description of the property link type	varchar(253)	253	True		
	PropertyLinkTypeName short name for the property link type	varchar(50)	50	True		
	HistoricPropertyLinkTypeID historic flg ID from access database	char(1)	1	True		
	ModifiedDate last modified date for the record	datetime	8	True		
	CreatedDate date the record was created	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_PropertyLinkType	PropertyLinkTypeID	True	UData

SQL Script

```
CREATE TABLE [dbo].[PropertyLinkType]
(
[PropertyLinkTypeID] [tinyint] NOT NULL IDENTITY(1, 1),
```

Author: liam

```
[PropertyLinkTypeDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS  
NULL,  
[PropertyLinkTypeName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,  
[HistoricPropertyLinkTypeID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,  
[ModifiedDate] [datetime] NULL,  
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PropertyLinkType_CreatedDate] DEFAULT  
(getdate())  
) ON [UData]  
GO  
ALTER TABLE [dbo].[PropertyLinkType] ADD CONSTRAINT [PK_PropertyLinkType] PRIMARY KEY  
CLUSTERED ([PropertyLinkTypeID]) ON [UData]  
GO  
EXEC sp_adddextendedproperty N'MS_Description', N'Collection of property link types',  
'SCHEMA', N'dbo', 'TABLE', N'PropertyLinkType', NULL, NULL  
GO  
EXEC sp_adddextendedproperty N'MS_Description', N'date the record was created',  
'SCHEMA', N'dbo', 'TABLE', N'PropertyLinkType', 'COLUMN', N'CreatedDate'  
GO  
EXEC sp_adddextendedproperty N'MS_Description', N'historic flg ID from access database',  
'SCHEMA', N'dbo', 'TABLE', N'PropertyLinkType', 'COLUMN', N'HistoricPropertyLinkTypeID'  
GO  
EXEC sp_adddextendedproperty N'MS_Description', N'last modified date for the record',  
'SCHEMA', N'dbo', 'TABLE', N'PropertyLinkType', 'COLUMN', N'ModifiedDate'  
GO  
EXEC sp_adddextendedproperty N'MS_Description', N'Detailed description of the property  
link type', 'SCHEMA', N'dbo', 'TABLE', N'PropertyLinkType', 'COLUMN', N'PropertyLink-  
TypeDescription'  
GO  
EXEC sp_adddextendedproperty N'MS_Description', N'short name for the property link  
type', 'SCHEMA', N'dbo', 'TABLE', N'PropertyLinkType', 'COLUMN', N'PropertyLinkType-  
Name'  
GO
```

Used By

[dbo].[FamilytoProperty]

[dbo].[PropertyNotes]

MS_Description

linking table for access agreement and access agreement notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	5439
Created	12:32:17 AM Tuesday, February 17, 2015
Last Modified	12:32:17 AM Tuesday, February 17, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	PropertyNotesID	int	4	False	1 - 1	
 D	PropertyID	int	4	False		
	CreatedDate date the notes were added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	False		

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_PropertyNotes	PropertyNotesID	True	UData

Foreign Keys

Name	Columns
FK_PropertyNotes_Property	PropertyID->[dbo].[Property].[PropertyID]

SQL Script

```
CREATE TABLE [dbo].[PropertyNotes]
(

```

Author: liam

```
[PropertyNotesID] [int] NOT NULL IDENTITY(1, 1),
[PropertyID] [int] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PropertyNotes_CreatedDate] DEFAULT
(getdate()),
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NOT NULL
) ON [UData]
GO
ALTER TABLE [dbo].[PropertyNotes] ADD CONSTRAINT [PK_PropertyNotes] PRIMARY KEY
CLUSTERED ([PropertyNotesID]) ON [UData]
GO
ALTER TABLE [dbo].[PropertyNotes] ADD CONSTRAINT [FK_PropertyNotes_Property] FOREIGN
KEY ([PropertyID]) REFERENCES [dbo].[Property] ([PropertyID])
GO
EXEC sp_adddextendedproperty N'MS_Description', N'linking table for access agreement and
access agreement notes', 'SCHEMA', N'dbo', 'TABLE', N'PropertyNotes', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the notes where added', 'SCHEMA',
N'dbo', 'TABLE', N'PropertyNotes', 'COLUMN', N'CreatedDate'
GO
```

Uses

[dbo].[Property]

Used By

[dbo].[usp_InsertPropertyNotes]

[dbo].[PropertySampleResults]	

MS_Description

collection of property test results

Properties

Property	Value
File Group	UData
Row Count (~)	4
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	12:21:37 PM Saturday, February 21, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	PropertySampleResultsID unique identifier for property test results	int	4	False	1 - 1	
	isBaseline is this a baseline test result for the property	bit	1	False		((0))
	PropertyID id of the property to which the test results apply	int	4	False		
	LabSubmissionDate date the property test samples were submitted to the lab	date	3	True		
	LabID id of the lab to which the property samples were submitted	int	4	True		
	SampleTypeID id of the sample type	tinyint	1	True		
	CreatedDate	datetime	8	True		(getdate())
	ModifiedDate	datetime	8	True		

Indexes

Key	Name	Columns	Unique	File Group
	PK_PropertySampleResults	PropertySample-ResultsID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdatePropertySampleResults	True	True	After Update

Check Constraints

Name	On Column	Constraint
ck_PropertySampleResults_LabSubmissionDate	LabSubmissionDate	([dbo].[udf_DateInThePast]([LabSubmissionDate])=(1))

Foreign Keys

Name	Columns
FK_PropertySampleResults_Property	PropertyID->[dbo].[Property].[PropertyID]
FK_PropertySampleResults_SampleType	SampleTypeID->[dbo].[SampleType].[SampleTypeID]

SQL Script

```

CREATE TABLE [dbo].[PropertySampleResults]
(
[PropertySampleResultsID] [int] NOT NULL IDENTITY(1, 1),
[isBaseline] [bit] NOT NULL CONSTRAINT [DF_PropertyTestResults_isBaseline] DEFAULT
((0)),
[PropertyID] [int] NOT NULL,
[LabSubmissionDate] [date] NULL,
[LabID] [int] NULL,
[SampleTypeID] [tinyint] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PropertySampleResults_CreatedDate] DEFAULT
(getdate()),
[ModifiedDate] [datetime] NULL
) ON [UData]
GO
create trigger [dbo].[trUpdatePropertySampleResults] on [dbo].[PropertySampleResults]
AFTER UPDATE
    as
        begin
            if @@rowcount = 0
                return
            if not update(ModifiedDate) update PropertySampleResults set ModifiedDate =
getdate() where PropertySampleResultsID in (select PropertySampleResultsID from
inserted)

        end
GO
ALTER TABLE [dbo].[PropertySampleResults] ADD CONSTRAINT [ck_PropertySampleResults_Lab-
SubmissionDate] CHECK (([dbo].[udf_DateInThePast]([LabSubmissionDate])=(1)))
GO
ALTER TABLE [dbo].[PropertySampleResults] ADD CONSTRAINT [PK_PropertySampleResults]
PRIMARY KEY CLUSTERED ([PropertySampleResultsID]) ON [UData]

```

```
GO
ALTER TABLE [dbo].[PropertySampleResults] ADD CONSTRAINT [FK_PropertySampleResults_Property] FOREIGN KEY ([PropertyID]) REFERENCES [dbo].[Property] ([PropertyID])
GO
ALTER TABLE [dbo].[PropertySampleResults] ADD CONSTRAINT [FK_PropertySampleResults_SampleType] FOREIGN KEY ([SampleTypeID]) REFERENCES [dbo].[SampleType] ([SampleTypeID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of property test results', 'SCHEMA', N'dbo', 'TABLE', N'PropertySampleResults', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'is this a baseline test result for the property', 'SCHEMA', N'dbo', 'TABLE', N'PropertySampleResults', 'COLUMN', N'isBaseline'
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the lab to which the property samples were submitted', 'SCHEMA', N'dbo', 'TABLE', N'PropertySampleResults', 'COLUMN', N'LabID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the proeprty test samples were submitted to the lab', 'SCHEMA', N'dbo', 'TABLE', N'PropertySampleResults', 'COLUMN', N'LabSubmissionDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the property to which the test results apply', 'SCHEMA', N'dbo', 'TABLE', N'PropertySampleResults', 'COLUMN', N'PropertyID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'unique identifier for property test results', 'SCHEMA', N'dbo', 'TABLE', N'PropertySampleResults', 'COLUMN', N'PropertySampleResultsID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the sample type', 'SCHEMA', N'dbo', 'TABLE', N'PropertySampleResults', 'COLUMN', N'SampleTypeID'
GO
```

Uses

[dbo].[Property]
[dbo].[SampleType]
[dbo].[udf_DateInThePast]

Used By

[dbo].[PropertySampleResultsNotes]
[dbo].[usp_InsertPropertySampleResults]

[dbo].[PropertySampleResultsNotes]	

MS_Description

linking table for access agreement and access agreement notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	1
Created	12:34:16 AM Tuesday, February 17, 2015
Last Modified	12:37:34 AM Tuesday, February 17, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	PropertySampleResultsNotesID	int	4	False	1 - 1	
	PropertySampleResultsID	int	4	False		
	CreatedDate date the notes where added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	False		

Indexes

Key	Name	Columns	Unique	File Group
	PK_PropertySampleResultsNotes	PropertySampleResultsNotesID	True	UData

Foreign Keys

Name	Columns
FK_PropertySampleResultsNotes_PropertySampleResults	PropertySampleResultsID->[dbo].[PropertySampleResults].[PropertySampleResultsID]

SQL Script

```
CREATE TABLE [dbo].[PropertySampleResultsNotes]
(
[PropertySampleResultsNotesID] [int] NOT NULL IDENTITY(1, 1),
[PropertySampleResultsID] [int] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PropertySampleResultsNotes_CreatedDate]
DEFAULT (getdate()),
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NOT NULL
) ON [UData]
GO
ALTER TABLE [dbo].[PropertySampleResultsNotes] ADD CONSTRAINT [PK_PropertySampleResults-
Notes] PRIMARY KEY CLUSTERED ([PropertySampleResultsNotesID]) ON [UData]
GO
ALTER TABLE [dbo].[PropertySampleResultsNotes] ADD CONSTRAINT [FK_PropertySampleResults-
Notes_PropertySampleResults] FOREIGN KEY ([PropertySampleResultsID]) REFERENCES
[dbo].[PropertySampleResults] ([PropertySampleResultsID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for access agreement and
access agreement notes', 'SCHEMA', N'dbo', 'TABLE', N'PropertySampleResultsNotes',
NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the notes where added', 'SCHEMA',
N'dbo', 'TABLE', N'PropertySampleResultsNotes', 'COLUMN', N'CreatedDate'
GO
```

Uses

[dbo].[PropertySampleResults]

Used By

[dbo].[usp_InsertPropertySampleResultsNotes]

[dbo].[PropertytoCleanupStatus]

MS_Description

linking table for property and cleanup status

Properties

Property	Value
File Group	UData
Row Count (~)	7
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	2:11:32 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PropertyID	int	4	False	
	CleanupStatusID	tinyint	1	False	
	CleanupStatusDate date of the cleanup status	date	3	False	(getdate())
	CostofCleanup cost of the cleanup	money	8	True	
	CreatedDate	datetime	8	True	(getdate())
	ModifiedDate	datetime	8	True	

Indexes

Key	Name	Columns	Unique	File Group
	PK_PropertytoCleanupStatus	PropertyID, CleanupStatusID, CleanupStatusDate	True	UData

Foreign Keys

Name	Columns
FK_PropertytoCleanupStatus_CleanupStatus	CleanupStatusID->[dbo].[CleanupStatus].[CleanupStatusID]
FK_PropertytoCleanupStatus_Property	PropertyID->[dbo].[Property].[PropertyID]

SQL Script

```
CREATE TABLE [dbo].[PropertytoCleanupStatus]
(
[PropertyID] [int] NOT NULL,
[CleanupStatusID] [tinyint] NOT NULL,
[CleanupStatusDate] [date] NOT NULL CONSTRAINT [DF_PropertytoCleanupStatus_Cleanup-
StatusDate] DEFAULT (getdate()),
[CostofCleanup] [money] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PropertytoCleanupStatus_CreatedDate]
DEFAULT (getdate()),
[ModifiedDate] [datetime] NULL
) ON [UData]
GO
ALTER TABLE [dbo].[PropertytoCleanupStatus] ADD CONSTRAINT [PK_PropertytoCleanupStatus]
PRIMARY KEY CLUSTERED ([PropertyID], [CleanupStatusID], [CleanupStatusDate]) ON
[UData]
GO
ALTER TABLE [dbo].[PropertytoCleanupStatus] ADD CONSTRAINT [FK_PropertytoCleanupStatus_-
CleanupStatus] FOREIGN KEY ([CleanupStatusID]) REFERENCES [dbo].[CleanupStatus]
([CleanupStatusID])
GO
ALTER TABLE [dbo].[PropertytoCleanupStatus] ADD CONSTRAINT [FK_PropertytoCleanupStatus_-
Property] FOREIGN KEY ([PropertyID]) REFERENCES [dbo].[Property] ([PropertyID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for property and cleanup
status', 'SCHEMA', N'dbo', 'TABLE', N'PropertytoCleanupStatus', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date of the cleanup status', 'SCHEMA',
N'dbo', 'TABLE', N'PropertytoCleanupStatus', 'COLUMN', N'CleanupStatusDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'cost of the cleanup', 'SCHEMA',
N'dbo', 'TABLE', N'PropertytoCleanupStatus', 'COLUMN', N'CostofCleanup'
GO
```

Uses

[dbo].[CleanupStatus]
[dbo].[Property]

Used By

[dbo].[usp_InsertPropertytoCleanupStatus]

[dbo].[PropertytoHouseholdSourcesofLead]

MS_Description

linking table for property and household sources of lead

Properties

Property	Value
File Group	UData
Row Count (~)	0
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	2:11:32 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PropertyID	int	4	False	
	HouseholdSourcesofLeadID	int	4	False	
	CreatedDate	datetime	8	True	(getdate())
	ModifiedDate	datetime	8	True	

Indexes

Key	Name	Columns	Unique	File Group
	PK_PropertytoHouseholdSourcesofLead	PropertyID, Household-SourcesofLeadID	True	UData

Foreign Keys

Name	Columns
FK_HouseholdSourcesofLead_PropertytoHousehold-SourcesofLead	HouseholdSourcesofLeadID->[dbo].[HouseholdSourcesofLead].[HouseholdSourcesofLeadID]
FK_Property_PropertytoHouseholdSourcesofLead	PropertyID->[dbo].[Property].[PropertyID]

SQL Script

```
CREATE TABLE [dbo].[PropertytoHouseholdSourcesofLead]
```

Author: liam

```
(  
    [PropertyID] [int] NOT NULL,  
    [HouseholdSourcesofLeadID] [int] NOT NULL,  
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_PropertytoHouseholdSourcesofLead_Created-  
    Date] DEFAULT (getdate()),  
    [ModifiedDate] [datetime] NULL  
) ON [UData]  
GO  
ALTER TABLE [dbo].[PropertytoHouseholdSourcesofLead] ADD CONSTRAINT [PK_Propertyto-  
HouseholdSourcesofLead] PRIMARY KEY CLUSTERED ([PropertyID], [HouseholdSourcesofLead-  
ID]) ON [UData]  
GO  
ALTER TABLE [dbo].[PropertytoHouseholdSourcesofLead] ADD CONSTRAINT [FK_Household-  
SourcesofLead_PropertytoHouseholdSourcesofLead] FOREIGN KEY ([HouseholdSourcesofLead-  
ID]) REFERENCES [dbo].[HouseholdSourcesofLead] ([HouseholdSourcesofLeadID])  
GO  
ALTER TABLE [dbo].[PropertytoHouseholdSourcesofLead] ADD CONSTRAINT [FK_Property_-  
PropertytoHouseholdSourcesofLead] FOREIGN KEY ([PropertyID]) REFERENCES  
[dbo].[Property] ([PropertyID])  
GO  
EXEC sp_addextendedproperty N'MS_Description', N'linking table for property and  
household sources of lead', 'SCHEMA', N'dbo', 'TABLE', N'PropertytoHouseholdSourcesof-  
Lead', NULL, NULL  
GO
```

Uses

[dbo].[HouseholdSourcesofLead]
[dbo].[Property]

Used By

[dbo].[usp_InsertPropertytoHouseholdSourcesofLead]

[dbo].[PropertytoMedium]

MS_Description

linking table for property and media

Properties

Property	Value
File Group	UData
Row Count (~)	1
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	2:11:32 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Default
	PropertyID	int	4	False	
	MediumID	int	4	False	
	MediumTested 0 - yes; 1 - no. Has the medium been tested.	bit	1	False	
	CreatedDate	datetime	8	True	(getdate())
	ModifiedDate	datetime	8	True	

Indexes

Key	Name	Columns	Unique	File Group
	PK_PropertytoMedium	PropertyID, MediumID	True	UData

Foreign Keys

Name	Columns
FK_PropertytoMedium_Medium	MediumID->[dbo].[Medium].[MediumID]
FK_PropertytoMedium_Property	PropertyID->[dbo].[Property].[PropertyID]

SQL Script

```
CREATE TABLE [dbo].[PropertytoMedium]
(
[PropertyID] [int] NOT NULL,
[MediumID] [int] NOT NULL,
[MediumTested] [bit] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_PropertytoMedium_CreatedDate] DEFAULT
(getdate()),
[ModifiedDate] [datetime] NULL
) ON [UData]
GO
ALTER TABLE [dbo].[PropertytoMedium] ADD CONSTRAINT [PK_PropertytoMedium] PRIMARY KEY
CLUSTERED ([PropertyID], [MediumID]) ON [UData]
GO
ALTER TABLE [dbo].[PropertytoMedium] ADD CONSTRAINT [FK_PropertytoMedium_Medium]
FOREIGN KEY ([MediumID]) REFERENCES [dbo].[Medium] ([MediumID])
GO
ALTER TABLE [dbo].[PropertytoMedium] ADD CONSTRAINT [FK_PropertytoMedium_Property]
FOREIGN KEY ([PropertyID]) REFERENCES [dbo].[Property] ([PropertyID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for property and media',
'SCHEMA', N'dbo', 'TABLE', N'PropertytoMedium', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 - yes; 1 - no. Has the medium been
tested.', 'SCHEMA', N'dbo', 'TABLE', N'PropertytoMedium', 'COLUMN', N'MediumTested'
GO
```

Uses

[dbo].[Medium]
[dbo].[Property]

Used By

[dbo].[usp_InsertPropertytoMedium]

[dbo].[Questionnaire]

MS_Description

collection of questionnaire questions and answers, typically only completed by flagged patients

Properties

Property	Value
File Group	UData
Row Count (~)	6156
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	1:05:24 PM Thursday, June 18, 2015

Columns

Key	Name	Data Type	Computed	Max Length (Bytes)	Allow Nulls	Identity	Default
	QuestionnaireID unique identifier for the questionnaire object	int		4	False	1 - 1	
(2)	PersonID id of the patient the questionnaire is referring to	int		4	False		
(2)	QuestionnaireDate Date the questionnaire was completed	date		3	True		
	QuestionnaireDataSource-ID id of the person completing the questionnaire	int		4	True		
	VisitRemodeledProperty 0 = no; 1 = yes. does the patient frequently visited remodeled properties	bit		1	True		
	isExposedtoPeelingPaint 0 = no; 1 = yes. has the patient been exposed to peeling paint	bit		1	True		
	isTakingVitamins 0 = no; 1 = yes. Is the patient taking vitamins regularly	bit		1	True		((0))
	NursingMother 0 = no; 1 = yes. is the patient a mother nursing a child	bit		1	True		((0))
	isUsingPacifier 0 = no; 1 = yes. is the	bit		1	True		((0))

	patient using a pacifier						
	isUsingBottle 0 = no; 1 = yes. is the patient using a bottle	bit		1	True		((0))
	BitesNails 0 = no; 1 = yes. does the patient bite nails	bit		1	True		((0))
	NonFoodEating 0 = no; 1 = yes. does the patient consume non food products	bit		1	True		((0))
	NonFoodinMouth 0 = no; 1 = yes. does the patient put non food items in mouth?	bit		1	True		((0))
	EatOutside 0 = no; 1 = yes. does the patient eat outside?	bit		1	True		((0))
	Suckling 0 = no; 1 = yes. does the patient suck his/her thumb or suckle	bit		1	True		((0))
	FrequentHandWashing 0 = no; 1 = yes. does the patient frequently wash hands throughout the day	bit		1	True		((0))
	DaycareID id of the daycare the patient attends	int		4	True		
	CreatedDate Date the record was created	datetime		8	True		(getdate())
	ModifiedDate date the record was last modified	datetime		8	True		
	ReviewStatusID Review Status ID	tinyint		1	True		
	Mouthing 0 = no; 1 = yes. does the client mouth things frequently	bit		1	True		
	VisitsOldHomes 0 = no; 1 = yes. does the patient visit older homes	bit		1	True		
	NursingInfant 0 = no; 1 = yes. is the patient a nursing infant	bit		1	True		
	Pregnant 0 = no; 1 = yes. is the patient pregnant	bit		1	True		
	PaintDate	date		3	True		
	RemodelPropertyDate	date		3	True		
	PaintAge	int	True	4	True		

	RemodelPropertyAge	int	True	4	True		
--	--------------------	-----	------	---	------	--	--

Computed columns

Name	Column definition
PaintAge	([dbo].[udf_CalculateAge]([PaintDate],getdate()))
RemodelPropertyAge	([dbo].[udf_CalculateAge]([RemodelPropertyDate],getdate()))

Indexes

Key	Name	Columns	Unique	Fill Factor	File Group
 PK_Questionnaire		QuestionnaireID	True		UData
	IDX_QuestionnaireDateIDPersonIDNursingMother-Pregnant	QuestionnaireID, PersonID, Nursing-Mother, Pregnant, QuestionnaireDate			UData
	NonClusteredIndex-PersonIDQuestionnaireDate	PersonID, QuestionnaireDate		90	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateQuestionnaire	True	True	After Update

Check Constraints

Name	On Column	Constraint
ck_Questionnaire_PaintDate	PaintDate	([dbo].[udf_DateInThePast]([PaintDate])=(1) OR [PaintDate] IS NULL)
ck_Questionnaire_QuestionnaireDate	QuestionnaireDate	([dbo].[udf_DateInThePast]([QuestionnaireDate])=(1))
ck_Questionnaire_RemodelPropertyDate	RemodelPropertyDate	([dbo].[udf_DateInThePast]([RemodelPropertyDate])=(1) OR [RemodelPropertyDate] IS NULL)

Foreign Keys

Name	Columns
FK_Questionnaire_Daycare	DaycareID->[dbo].[Daycare].[DaycareID]
FK_Questionnaire_Person	PersonID->[dbo].[Person].[PersonID]

FK_Questionnaire_ReviewStatus	ReviewStatusID->[dbo].[ReviewStatus].[ReviewStatusID]
-------------------------------	---

SQL Script

```

CREATE TABLE [dbo].[Questionnaire]
(
    [QuestionnaireID] [int] NOT NULL IDENTITY(1, 1),
    [PersonID] [int] NOT NULL,
    [QuestionnaireDate] [date] NULL,
    [QuestionnaireDataSourceID] [int] NULL,
    [VisitRemodeledProperty] [bit] NULL,
    [isExposedtoPeelingPaint] [bit] NULL,
    [isTakingVitamins] [bit] NULL CONSTRAINT [DF_Questionnaire_isTakingVitamins] DEFAULT ((0)),
    [NursingMother] [bit] NULL CONSTRAINT [DF_Questionnaire_isNursing] DEFAULT ((0)),
    [isUsingPacifier] [bit] NULL CONSTRAINT [DF_Questionnaire_isUsingPacifier] DEFAULT ((0)),
    [isUsingBottle] [bit] NULL CONSTRAINT [DF_Questionnaire_isUsingBottle] DEFAULT ((0)),
    [BitesNails] [bit] NULL CONSTRAINT [DF_Questionnaire_Bitesnails] DEFAULT ((0)),
    [NonFoodEating] [bit] NULL CONSTRAINT [DF_Questionnaire_NonFoodEating] DEFAULT ((0)),
    [NonFoodinMouth] [bit] NULL CONSTRAINT [DF_Questionnaire_NonFoodinMouth] DEFAULT ((0)),
    [EatOutside] [bit] NULL CONSTRAINT [DF_Questionnaire_EatOutside] DEFAULT ((0)),
    [Suckling] [bit] NULL CONSTRAINT [DF_Questionnaire_Suckling] DEFAULT ((0)),
    [FrequentHandWashing] [bit] NULL CONSTRAINT [DF_Questionnaire_FrequentHandWashing] DEFAULT ((0)),
    [DaycareID] [int] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_Questionnaire_CreatedDate] DEFAULT (getdate()),
    [ModifiedDate] [datetime] NULL,
    [ReviewStatusID] [tinyint] NULL,
    [Mouthing] [bit] NULL,
    [VisitsOldHomes] [bit] NULL,
    [NursingInfant] [bit] NULL,
    [Pregnant] [bit] NULL,
    [PaintDate] [date] NULL,
    [RemodelPropertyDate] [date] NULL,
    [PaintAge] AS ([dbo].[udf_CalculateAge]([PaintDate],getdate())),
    [RemodelPropertyAge] AS ([dbo].[udf_CalculateAge]([RemodelPropertyDate],getdate()))
) ON [UData]
GO
create trigger [dbo].[trUpdateQuestionnaire] on [dbo].[Questionnaire] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update Questionnaire set ModifiedDate = getdate()
    where QuestionnaireID in (select QuestionnaireID from inserted)

    end
GO
ALTER TABLE [dbo].[Questionnaire] ADD CONSTRAINT [ck_Questionnaire_PaintDate] CHECK
(([dbo].[udf_DateInThePast]([PaintDate])=(1) OR [PaintDate] IS NULL))
GO

```

Project> (local)> User databases> LCCHPDev> Tables> dbo.Questionnaire

```

ALTER TABLE [dbo].[Questionnaire] ADD CONSTRAINT [ck_Questionnaire_QuestionnaireDate]
CHECK (([dbo].[udf_DateInThePast]([QuestionnaireDate])=(1)))
GO
ALTER TABLE [dbo].[Questionnaire] ADD CONSTRAINT [ck_Questionnaire_RemodelPropertyDate]
CHECK (((dbo].[udf_DateInThePast]([RemodelPropertyDate])=(1) OR [RemodelPropertyDate]
IS NULL))
GO
ALTER TABLE [dbo].[Questionnaire] ADD CONSTRAINT [PK_Questionnaire] PRIMARY KEY
CLUSTERED ([QuestionnaireID]) ON [UData]
GO
CREATE NONCLUSTERED INDEX [NonClusteredIndex-PersonIDQuestionnaireDate] ON
[dbo].[Questionnaire] ([PersonID], [QuestionnaireDate]) ON [UData]
GO
CREATE NONCLUSTERED INDEX [IDX_QuestionnaireDateIDPersonIDNursingMotherPregnant] ON
[dbo].[Questionnaire] ([QuestionnaireDate]) INCLUDE ([NursingMother], [PersonID],
[Pregnant], [QuestionnaireID]) ON [UData]
GO
ALTER TABLE [dbo].[Questionnaire] ADD CONSTRAINT [FK_Questionnaire_Daycare] FOREIGN KEY
([DaycareID]) REFERENCES [dbo].[Daycare] ([DaycareID])
GO
ALTER TABLE [dbo].[Questionnaire] ADD CONSTRAINT [FK_Questionnaire_Person] FOREIGN KEY
([PersonID]) REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Questionnaire] ADD CONSTRAINT [FK_Questionnaire_ReviewStatus]
FOREIGN KEY ([ReviewStatusID]) REFERENCES [dbo].[ReviewStatus] ([ReviewStatusID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of questionnaire questions
and answers, typically only completed by flagged patients', 'SCHEMA', N'dbo', 'TABLE',
N'Questionnaire', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. does the patient
bite nails', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'BitesNails'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Date the record was created',
'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'CreatedDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the daycare the patient
attends', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'DaycareID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. does the patient eat
outside?', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'EatOutside'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. does the patient frequently
wash hands throughout the day', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire',
'COLUMN', N'FrequentHandWashing'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. has the patient been
exposed to peeling paint', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'is-
ExposedtoPeelingPaint'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. Is the patient
taking vitamins regularly', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'is-
TakingVitamins'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. is the patient using
a bottle', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'isUsingBottle'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. is the patient using
a pacifier', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'isUsingPacifier'

```

Author: liam

```

GO
EXEC sp_addextendedproperty N'MS_Description', N'date the record was last modified',
'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'ModifiedDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. does the client
mouth things frequently', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN',
N'Mouthing'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. does the patient
consume non food products', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN',
N'NonFoodEating'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. does the patient put
non food items in mouth?', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'Non-
FoodinMouth'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. is the patient a
nursing infant', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'Nursing-
Infant'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. is the patient a
mother nursing a child', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN',
N'NursingMother'
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the patient the questionnaire is
referring to', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'PersonID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. is the patient
pregnant', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'Pregnant'
GO
EXEC sp_addextendedproperty N'MS_Description', N'id of the person completing the
questionnaire', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'Questionnaire-
DataSourceID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Date the questionnaire was completed',
'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'QuestionnaireDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'unique identifier for the
questionnaire object', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN',
N'QuestionnaireID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Review Status ID', 'SCHEMA', N'dbo',
'TABLE', N'Questionnaire', 'COLUMN', N'ReviewStatusID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. does the patient
suck his/her thumb or suckle', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN',
N'Suckling'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. does the patient
frequently visited remodeled properties', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire',
'COLUMN', N'VisitRemodeledProperty'
GO
EXEC sp_addextendedproperty N'MS_Description', N'0 = no; 1 = yes. does the patient
visit older homes', 'SCHEMA', N'dbo', 'TABLE', N'Questionnaire', 'COLUMN', N'VisitsOld-
Homes'
GO

```

Uses

[dbo].[Daycare]
[dbo].[Person]
[dbo].[ReviewStatus]
[dbo].[udf_CalculateAge]
[dbo].[udf_DateInThePast]

Used By

[dbo].[QuestionnaireNotes]
[dbo].[vMostRecentQuestionnaires]
[dbo].[usp_InsertQuestionnaire]
[dbo].[usp_SICountAdults]
[dbo].[usp_SIISummaryReport]
[dbo].[usp_upClientFlag]
[dbo].[usp_upQuestionnaire]

[dbo].[QuestionnaireDataSource]

MS_Description

source of the data (Environmental group or Blood Lead)

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	2
Created	7:58:38 PM Sunday, April 19, 2015
Last Modified	8:01:47 PM Sunday, April 19, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity
	QuestionnaireDataSourceID unique identifier for the questionnaire data source	int	4	False	1 - 1
	QuestionnaireDataSourceName Source of the questionnaire data - environmental or blood lead	varchar(50)	50	False	
	QuestionnaireDataSourceDescription More details about the source of the questionnaire data - environmental or blood lead	varchar(253)	253	True	

Indexes

Key	Name	Columns	Unique	File Group
	PK_QuestionnaireDataSource	QuestionnaireDataSourceID	True	UData

SQL Script

```
CREATE TABLE [dbo].[QuestionnaireDataSource]
(
    [QuestionnaireDataSourceID] [int] NOT NULL IDENTITY(1, 1),
    [QuestionnaireDataSourceName] [varchar](50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
    [QuestionnaireDataSourceDescription] [varchar](253) COLLATE SQL_Latin1_General_CI_AS NULL
) ON [UData]
```

```
GO
ALTER TABLE [dbo].[QuestionnaireDataSource] ADD CONSTRAINT [PK_QuestionnaireDataSource]
PRIMARY KEY CLUSTERED ([QuestionnaireDataSourceID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'source of the data (Environmental
group or Blood Lead)', 'SCHEMA', N'dbo', 'TABLE', N'QuestionnaireDataSource', NULL,
NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'More details about the source of the
questionnaire data - enviornmental or blood lead', 'SCHEMA', N'dbo', 'TABLE',
N'QuestionnaireDataSource', 'COLUMN', N'QuestionnaireDataSourceDescription'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'unique identifier for the
questionnaire data source', 'SCHEMA', N'dbo', 'TABLE', N'QuestionnaireDataSource',
'COLUMN', N'QuestionnaireDataSourceID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Source of the questionnaire data -
enviornmental or blood lead', 'SCHEMA', N'dbo', 'TABLE', N'QuestionnaireDataSource',
'COLUMN', N'QuestionnaireDataSourceName'
GO
```

[dbo].[QuestionnaireNotes]

MS_Description

linking table for access agreement and access agreement notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	1457
Created	12:36:12 AM Tuesday, February 17, 2015
Last Modified	12:36:12 AM Tuesday, February 17, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	QuestionnaireNotesID	int	4	False	1 - 1	
 D	QuestionnaireID	int	4	False		
	CreatedDate date the notes where added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	False		

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_QuestionnaireNotes	QuestionnaireNotesID	True	UData

Foreign Keys

Name	Columns
FK_QuestionnaireNotes_Questionnaire	QuestionnaireID->[dbo].[Questionnaire].[QuestionnaireID]

SQL Script

```
CREATE TABLE [dbo].[QuestionnaireNotes]
(
```

Author: liam

```
[QuestionnaireNotesID] [int] NOT NULL IDENTITY(1, 1),
[QuestionnaireID] [int] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_QuestionnaireNotes_CreatedDate] DEFAULT
(getdate()),
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NOT NULL
) ON [UData]
GO
ALTER TABLE [dbo].[QuestionnaireNotes] ADD CONSTRAINT [PK_QuestionnaireNotes] PRIMARY
KEY CLUSTERED ([QuestionnaireNotesID]) ON [UData]
GO
ALTER TABLE [dbo].[QuestionnaireNotes] ADD CONSTRAINT [FK_QuestionnaireNotes_-
Questionnaire] FOREIGN KEY ([QuestionnaireID]) REFERENCES [dbo].[Questionnaire]
([QuestionnaireID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'linking table for access agreement and
access agreement notes', 'SCHEMA', N'dbo', 'TABLE', N'QuestionnaireNotes', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the notes where added', 'SCHEMA',
N'dbo', 'TABLE', N'QuestionnaireNotes', 'COLUMN', N'CreatedDate'
GO
```

Uses

[dbo].[Questionnaire]

Used By

[dbo].[usp_InsertQuestionnaireNotes]

[dbo].[RelationshipType]

MS_Description

collection of RelationshipType names and basic attributes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	24
Created	5:38:08 PM Saturday, January 3, 2015
Last Modified	10:11:41 PM Saturday, January 3, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	RelationshipTypeID unique identifier for the Relationship-Type object	int	4	False	1 - 1	
	RelationshipTypeName	varchar(50)	50	True		
	RelationshipTypeDescription	varchar(253)	253	True		
	CreatedDate	datetime	8	True		(getdate())
	ModifiedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_RelationshipType	RelationshipTypeID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateRelationshipType	True	True	After Update

SQL Script

```
CREATE TABLE [dbo].[RelationshipType]
(
[RelationshipTypeID] [int] NOT NULL IDENTITY(1, 1),
[RelationshipTypeName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[RelationshipTypeDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_RelationshipType_CreatedDate] DEFAULT (getdate()),
[ModifiedDate] [datetime] NULL CONSTRAINT [DF_RelationshipType_ModifiedDate] DEFAULT (getdate())
) ON [UData]
GO

create trigger [dbo].[trUpdateRelationshipType] on [dbo].[RelationshipType] AFTER
UPDATE
as
begin
if @@rowcount = 0
    return
    if not update(ModifiedDate) update RelationshipType set ModifiedDate = getdate()
where RelationshipTypeID in (select RelationshipTypeID from inserted)

end
GO
ALTER TABLE [dbo].[RelationshipType] ADD CONSTRAINT [PK_RelationshipType] PRIMARY KEY
CLUSTERED ([RelationshipTypeID]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of RelationshipType names
and basic attributes', 'SCHEMA', 'dbo', 'TABLE', N'RelationshipType', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'unique identifier for the Relationship-
Type object', 'SCHEMA', N'dbo', 'TABLE', N'RelationshipType', 'COLUMN', N'Relationship-
TypeID'
GO
```

Used By

[dbo].[PersontoPerson]

[dbo].[ReleaseStatus]

MS_Description

Collection of Release Status

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	0
Created	4:56:45 PM Saturday, April 11, 2015
Last Modified	2:17:22 PM Sunday, April 19, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	ReleaseStatusID	tinyint	1	False	1 - 1	
	ReleaseStatusDescription Detailed description of the Release status	varchar(253)	253	True		
	ReleaseStatusName short name for the Release status	varchar(50)	50	True		
	HistoricReleaseStatusID historic ReleaseStatus ID from access database	char(1)	1	True		
	ModifiedDate last modified date for the record	datetime	8	True		
	CreatedDate date the record was created	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_ReleaseStatus	ReleaseStatusID	True	UData

SQL Script

```
CREATE TABLE [dbo].[ReleaseStatus]
(
```

Author: liam

```
[ReleaseStatusID] [tinyint] NOT NULL IDENTITY(1, 1),
[ReleaseStatusDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[ReleaseStatusName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[HistoricReleaseStatusID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_ReleaseStatus_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[ReleaseStatus] ADD CONSTRAINT [PK_ReleaseStatus] PRIMARY KEY
CLUSTERED ([ReleaseStatusID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Collection of Release Status',
'SCHEMA', N'dbo', 'TABLE', N'ReleaseStatus', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the record was created',
'SCHEMA', N'dbo', 'TABLE', N'ReleaseStatus', 'COLUMN', N'CreatedDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'historic ReleaseStatus ID from access
database', 'SCHEMA', N'dbo', 'TABLE', N'ReleaseStatus', 'COLUMN', N'HistoricRelease-
StatusID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'last modified date for the record',
'SCHEMA', N'dbo', 'TABLE', N'ReleaseStatus', 'COLUMN', N'ModifiedDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Detailed description of the Release
status', 'SCHEMA', N'dbo', 'TABLE', N'ReleaseStatus', 'COLUMN', N'ReleaseStatus-
Description'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'short name for the Release status',
'SCHEMA', N'dbo', 'TABLE', N'ReleaseStatus', 'COLUMN', N'ReleaseStatusName'
GO
```

Used By

[dbo].[Property]

[dbo].[Remediation]	

MS_Description

collection of remediation data

Properties

Property	Value
File Group	UData
Row Count (~)	8
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	12:31:41 PM Saturday, February 21, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	RemediationID	int	4	False	1 - 1	
	RemediationApprovalDate	date	3	True		
	RemediationStartDate	date	3	True		
	RemediationEndDate	date	3	True		
	PropertyID	int	4	True		
	AccessAgreementID	int	4	True		
	FinalRemediationReportFile	varbinary(max)	max	True		
	FinalRemediationReportDate	date	3	True		
	RemediationCost	money	8	True		
	OneYearRemediationComplete-Date	date	3	True		
	OneYearRemediationComplete	bit	1	True		
	RemediationActionPlanID	int	4	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_Remediation	RemediationID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateRemediation	True	True	After Update

Check Constraints

Name	On Column	Constraint
ck_Remediation_RemediationApprovalDate	RemediationApprovalDate	([dbo].[udf_DateInThe-Past]([RemediationApprovalDate])=(1))

Foreign Keys

Name	Columns
FK_Remediation_Property	PropertyID->[dbo].[Property].[PropertyID]
FK_Remediation_RemediationActionPlan	RemediationActionPlanID->[dbo].[RemediationAction-Plan].[RemediationActionPlanID]

SQL Script

```

CREATE TABLE [dbo].[Remediation]
(
    [RemediationID] [int] NOT NULL IDENTITY(1, 1),
    [RemediationApprovalDate] [date] NULL,
    [RemediationStartDate] [date] NULL,
    [RemediationEndDate] [date] NULL,
    [PropertyID] [int] NULL,
    [AccessAgreementID] [int] NULL,
    [FinalRemediationReportFile] [varbinary] (max) NULL,
    [FinalRemediationReportDate] [date] NULL,
    [RemediationCost] [money] NULL,
    [OneYearRemediationCompleteDate] [date] NULL,
    [OneYearRemediationComplete] [bit] NULL,
    [RemediationActionPlanID] [int] NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_Remediation_CreatedDate] DEFAULT
    (getdate())
) ON [UData] TEXTIMAGE_ON [UData]
GO
create trigger [dbo].[trUpdateRemediation] on [dbo].[Remediation] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update Remediation set ModifiedDate = getdate() where
    RemediationID in (select RemediationID from inserted)

end

```

```
GO
ALTER TABLE [dbo].[Remediation] ADD CONSTRAINT [ck_Remediation_RemediationApprovalDate]
CHECK (([dbo].[udf_DateInThePast]([RemediationApprovalDate])=(1)))
GO
ALTER TABLE [dbo].[Remediation] ADD CONSTRAINT [PK_Remediation] PRIMARY KEY CLUSTERED
([RemediationID]) ON [UData]
GO
ALTER TABLE [dbo].[Remediation] ADD CONSTRAINT [FK_Remediation_Property] FOREIGN KEY
([PropertyID]) REFERENCES [dbo].[Property] ([PropertyID])
GO
ALTER TABLE [dbo].[Remediation] ADD CONSTRAINT [FK_Remediation_RemediationActionPlan]
FOREIGN KEY ([RemediationActionPlanID]) REFERENCES [dbo].[RemediationActionPlan]
([RemediationActionPlanID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of remediation data',
'SCHEMA', N'dbo', 'TABLE', N'Remediation', NULL, NULL
GO
```

Uses

[dbo].[Property]
[dbo].[RemediationActionPlan]
[dbo].[udf_DateInThePast]

Used By

[dbo].[ContractortoRemediation]
[dbo].[RemediationNotes]
[dbo].[usp_InsertRemediation]

[dbo].[RemediationActionPlan]

MS_Description

collection of sampling plans

Properties

Property	Value
File Group	UData
Row Count (~)	3
Created	1:12:50 AM Saturday, December 27, 2014
Last Modified	12:32:21 PM Saturday, February 21, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	RemediationActionPlanID	int	4	False	1 - 1	
	RemediationActionPlanApproval-Date	date	3	True		
	HomeOwnerConsultationDate Meeting date between homeowner and workgroup to review the sampling plan	date	3	True		
	ContractorCompletedInvestigation-Date	date	3	True		
	RemediationActionPlanFinalReport-SubmissionDate	date	3	True		
	RemediationActionPlanFile	varbinary(max)	max	True		
	PropertyID	int	4	True		
	EnvironmentalInvestigationID	int	4	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
 PK_RemediationActionPlan		RemediationActionPlan-ID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateRemediationActionPlan	True	True	After Update

Check Constraints

Name	On Column	Constraint
ck_RemediationActionPlan_RemediationActionPlanApprovalDate	RemediationAction-PlanApprovalDate	([dbo].[udf_DateIn-The-Past]([RemediationActionPlanApprovalDate])=(1))

Foreign Keys

Name	Columns
FK_RemediationActionPlan_EnvironmentalInvestigation	EnvironmentalInvestigationID->[dbo].[Environmental-Investigation].[EnvironmentalInvestigationID]

SQL Script

```

CREATE TABLE [dbo].[RemediationActionPlan]
(
    [RemediationActionPlanID] [int] NOT NULL IDENTITY(1, 1),
    [RemediationActionPlanApprovalDate] [date] NULL,
    [HomeOwnerConsultationDate] [date] NULL,
    [ContractorCompletedInvestigationDate] [date] NULL,
    [RemediationActionPlanFinalReportSubmissionDate] [date] NULL,
    [RemediationActionPlanFile] [varbinary] (max) NULL,
    [PropertyID] [int] NULL,
    [EnvironmentalInvestigationID] [int] NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_RemediationActionPlan_CreatedDate] DEFAULT
    (getdate())
) ON [UData] TEXTIMAGE_ON [UData]
GO
create trigger [dbo].[trUpdateRemediationActionPlan] on [dbo].[RemediationActionPlan]
AFTER UPDATE
as
begin
    if @@rowcount = 0

```

```
        return
    if not update(ModifiedDate) update RemediationActionPlan set ModifiedDate =
getdate() where RemediationActionPlanID in (select RemediationActionPlanID from
inserted)

    end
GO
ALTER TABLE [dbo].[RemediationActionPlan] ADD CONSTRAINT [ck_RemediationActionPlan_-
RemediationActionPlanApprovalDate] CHECK (([dbo].[udf_DateInThePast]([RemediationAction-
PlanApprovalDate])=(1)))
GO
ALTER TABLE [dbo].[RemediationActionPlan] ADD CONSTRAINT [PK_RemediationActionPlan]
PRIMARY KEY CLUSTERED ([RemediationActionPlanID]) ON [UData]
GO
ALTER TABLE [dbo].[RemediationActionPlan] ADD CONSTRAINT [FK_RemediationActionPlan_-
EnvironmentalInvestigation] FOREIGN KEY ([EnvironmentalInvestigationID]) REFERENCES
[dbo].[EnvironmentalInvestigation] ([EnvironmentalInvestigationID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of sampling plans',
'SCHEMA', N'dbo', 'TABLE', N'RemediationActionPlan', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'Meeting date between homeowner and
workgroup to review the sampling plan', 'SCHEMA', N'dbo', 'TABLE', N'RemediationAction-
Plan', 'COLUMN', N'HomeOwnerConsultationDate'
GO
```

Uses

[dbo].[EnvironmentalInvestigation]
[dbo].[udf_DateInThePast]

Used By

[dbo].[ContractorToRemediationActionPlan]
[dbo].[Remediation]
[dbo].[usp_InsertRemediationActionPlan]

[dbo].[RemediationNotes]

MS_Description

table for remediation notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	1
Created	12:41:02 AM Tuesday, February 17, 2015
Last Modified	12:41:02 AM Tuesday, February 17, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
 C	RemediationNotesID	int	4	False	1 - 1	
 D	RemediationID	int	4	False		
	CreatedDate date the notes where added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	False		

Indexes

Key	Name	Columns	Unique	File Group
 C	PK_RemediationNotes	RemediationNotesID	True	UData

Foreign Keys

Name	Columns
FK_RemediationNotes_Remediation	RemediationID->[dbo].[Remediation].[RemediationID]

SQL Script

```
CREATE TABLE [dbo].[RemediationNotes]
(
```

Author: liam

```
[RemediationNotesID] [int] NOT NULL IDENTITY(1, 1),
[RemediationID] [int] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_RemediationNotes_CreatedDate] DEFAULT
(getdate()),
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NOT NULL
) ON [UData]
GO
ALTER TABLE [dbo].[RemediationNotes] ADD CONSTRAINT [PK_RemediationNotes] PRIMARY KEY
CLUSTERED ([RemediationNotesID]) ON [UData]
GO
ALTER TABLE [dbo].[RemediationNotes] ADD CONSTRAINT [FK_RemediationNotes_Remediation]
FOREIGN KEY ([RemediationID]) REFERENCES [dbo].[Remediation] ([RemediationID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'table for remediation notes',
'SCHEMA', N'dbo', 'TABLE', N'RemediationNotes', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the notes were added', 'SCHEMA',
N'dbo', 'TABLE', N'RemediationNotes', 'COLUMN', N'CreatedDate'
GO
```

Uses

[dbo].[Remediation]

Used By

[dbo].[usp_InsertRemediationNotes]

[dbo].[ReviewStatus]	

MS_Description

Collection of potential status for Review

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	0
Created	5:46:41 PM Saturday, April 11, 2015
Last Modified	8:53:57 PM Sunday, April 19, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	ReviewStatusID	tinyint	1	False	1 - 1	
	ReviewStatusDescription Detailed description of the Review status	varchar(253)	253	True		
	ReviewStatusName status for the Review	varchar(50)	50	True		
	HistoricReviewStatusID historic status from access database	char(1)	1	True		
	ModifiedDate last modified date for the record	datetime	8	True		
	CreatedDate date the record was created	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_ReviewStatus	ReviewStatusID	True	UData

SQL Script

```
CREATE TABLE [dbo].[ReviewStatus]
(
[ReviewStatusID] [tinyint] NOT NULL IDENTITY(1, 1),
```

Author: liam

```
[ReviewStatusDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[ReviewStatusName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[HistoricReviewStatusID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_ReviewStatus_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[ReviewStatus] ADD CONSTRAINT [PK_ReviewStatus] PRIMARY KEY CLUSTERED
([ReviewStatusID]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'Collection of potential status for
Review', 'SCHEMA', N'dbo', 'TABLE', N'ReviewStatus', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the record was created',
'SCHEMA', N'dbo', 'TABLE', N'ReviewStatus', 'COLUMN', N'CreatedDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'historic status from access database',
'SCHEMA', N'dbo', 'TABLE', N'ReviewStatus', 'COLUMN', N'HistoricReviewStatusID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'last modified date for the record',
'SCHEMA', N'dbo', 'TABLE', N'ReviewStatus', 'COLUMN', N'ModifiedDate'
GO
EXEC sp_addextendedproperty N'MS_Description', N'Detailed description of the Review
status', 'SCHEMA', N'dbo', 'TABLE', N'ReviewStatus', 'COLUMN', N'ReviewStatus-
Description'
GO
EXEC sp_addextendedproperty N'MS_Description', N'status for the Review', 'SCHEMA',
N'dbo', 'TABLE', N'ReviewStatus', 'COLUMN', N'ReviewStatusName'
GO
```

Used By

[dbo].[FamilytoProperty]
[dbo].[Person]
[dbo].[Questionnaire]

[dbo].[SampleLevelCategory]

MS_Description

collection of sample level categorizations

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	1
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	8:24:48 PM Wednesday, March 4, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	SampleLevelCategoryID unique identifier for sample level categorization	tinyint	1	False	1 - 1	
	SampleLevelCategoryName description of sample level category	varchar(50)	50	True		
	SampleLevelCategoryDescription	varchar(253)	253	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_SampleLevelCategory	SampleLevelCategory-ID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateSampleLevelCategory	True	True	After Update

SQL Script

```

CREATE TABLE [dbo].[SampleLevelCategory]
(
    [SampleLevelCategoryID] [tinyint] NOT NULL IDENTITY(1, 1),
    [SampleLevelCategoryName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
    [SampleLevelCategoryDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_SampleLevelCategory_CreatedDate] DEFAULT
    (getdate())
) ON [UData]
GO
create trigger [dbo].[trUpdateSampleLevelCategory] on [dbo].[SampleLevelCategory] AFTER
UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update SampleLevelCategory set ModifiedDate =
    getdate() where SampleLevelCategoryID in (select SampleLevelCategoryID from inserted)

end
GO
ALTER TABLE [dbo].[SampleLevelCategory] ADD CONSTRAINT [PK_SampleLevelCategory] PRIMARY
KEY CLUSTERED ([SampleLevelCategoryID]) ON [UData]
GO
EXEC sp_addextendedproperty N'MS_Description', N'collection of sample level
categorizations', 'SCHEMA', 'dbo', 'TABLE', N'SampleLevelCategory', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'unique identifier for sample level
categorization', 'SCHEMA', N'dbo', 'TABLE', N'SampleLevelCategory', 'COLUMN', N'Sample-
LevelCategoryID'
GO
EXEC sp_addextendedproperty N'MS_Description', N'description of sample level category',
'SCHEMA', N'dbo', 'TABLE', N'SampleLevelCategory', 'COLUMN', N'SampleLevelCategoryName'
GO

```

Used By

[\[dbo\].\[BloodTestResults\]](#)
[\[dbo\].\[MediumSampleResults\]](#)
[\[dbo\].\[usp_InsertSampleLevelCategory\]](#)

[dbo].[SamplePurpose]

MS_Description

Collection of sample purposes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	0
Created	5:07:52 PM Saturday, April 11, 2015
Last Modified	5:07:52 PM Saturday, April 11, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	SamplePurposeID	tinyint	1	False	1 - 1	
	SamplePurposeDescription Detailed description of the sample purpose	varchar(253)	253	True		
	SamplePurposeName short name for the sample purpose	varchar(50)	50	True		
	HistoricSamplePurposeID historic SamplePurpose ID from access database	char(1)	1	True		
	ModifiedDate last modified date for the record	datetime	8	True		
	CreatedDate date the record was created	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_SamplePurpose	SamplePurposeID	True	UData

SQL Script

```
CREATE TABLE [dbo].[SamplePurpose]
(
```

Author: liam

Project> (local)> User databases> LCCHPDev> Tables> dbo.SamplePurpose

```
[SamplePurposeID] [tinyint] NOT NULL IDENTITY(1, 1),
[SamplePurposeDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[SamplePurposeName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[HistoricSamplePurposeID] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_SamplePurpose_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
ALTER TABLE [dbo].[SamplePurpose] ADD CONSTRAINT [PK_SamplePurpose] PRIMARY KEY
CLUSTERED ([SamplePurposeID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Collection of sample purposes',
'SCHEMA', N'dbo', 'TABLE', N'SamplePurpose', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'date the record was created',
'SCHEMA', N'dbo', 'TABLE', N'SamplePurpose', 'COLUMN', N'CreatedDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'historic SamplePurpose ID from access
database', 'SCHEMA', N'dbo', 'TABLE', N'SamplePurpose', 'COLUMN', N'HistoricSample-
PurposeID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'last modified date for the record',
'SCHEMA', N'dbo', 'TABLE', N'SamplePurpose', 'COLUMN', N'ModifiedDate'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'Detailed description of the sample
purpose', 'SCHEMA', N'dbo', 'TABLE', N'SamplePurpose', 'COLUMN', N'SamplePurpose-
Description'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'short name for the sample purpose',
'SCHEMA', N'dbo', 'TABLE', N'SamplePurpose', 'COLUMN', N'SamplePurposeName'
GO
```

[dbo].[SampleType]

MS_Description

collection of sample types

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	11
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	8:24:48 PM Wednesday, March 4, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	SampleTypeID unique identifier of sample type	tinyint	1	False	1 - 1	
	SampleTypeName friendly name for the sample type	varchar(50)	50	True		
	SampleTypeDescription extended description of the sample type	varchar(253)	253	True		
	historicSampleType	char(1)	1	True		
	SampleTarget	varchar(50)	50	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())

Indexes

Key	Name	Columns	Unique	File Group
	PK_SampleType	SampleTypeID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On

Author: liam

trUpdateSampleType	True	True	After Update
--------------------	------	------	--------------

SQL Script

```

CREATE TABLE [dbo].[SampleType]
(
[SampleTypeID] [tinyint] NOT NULL IDENTITY(1, 1),
[SampleTypeName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[SampleTypeDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[historicSampleType] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL,
[SampleTarget] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_SampleType_CreatedDate] DEFAULT
(getdate())
) ON [UData]
GO
create trigger [dbo].[trUpdateSampleType] on [dbo].[SampleType] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update SampleType set ModifiedDate = getdate() where
SampleTypeID in (select SampleTypeID from inserted)

    end
GO
ALTER TABLE [dbo].[SampleType] ADD CONSTRAINT [PK_SampleType] PRIMARY KEY CLUSTERED
([SampleTypeID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'collection of sample types', 'SCHEMA',
N'dbo', 'TABLE', N'SampleType', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'extended description of the sample
type', 'SCHEMA', N'dbo', 'TABLE', N'SampleType', 'COLUMN', N'SampleTypeDescription'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'unique identifier of sample type',
'SCHEMA', N'dbo', 'TABLE', N'SampleType', 'COLUMN', N'SampleTypeID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'friendly name for the sample type',
'SCHEMA', N'dbo', 'TABLE', N'SampleType', 'COLUMN', N'SampleTypeName'
GO

```

Used By

[dbo].[BloodTestResults]
[dbo].[PropertySampleResults]
[dbo].[usp_InsertSampleType]

[dbo].[Source]	

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	2
Created	8:27:57 PM Tuesday, January 27, 2015
Last Modified	8:27:57 PM Tuesday, January 27, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity
	SourceID	int	4	False	1 - 1
	SourceName	varchar(50)	50	False	
	SourceDescription	varchar(253)	253	True	

Indexes

Key	Name	Columns	Unique	File Group
	PK_Source	SourceID	True	UData

SQL Script

```

CREATE TABLE [dbo].[Source]
(
[SourceID] [int] NOT NULL IDENTITY(1, 1),
[SourceName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
[SourceDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL
) ON [UData]
GO
ALTER TABLE [dbo].[Source] ADD CONSTRAINT [PK_Source] PRIMARY KEY CLUSTERED ([SourceID]) ON [UData]
GO

```

[dbo].[TargetStatus]

MS_Description

collection of status objects

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	22
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	5:31:42 PM Thursday, June 11, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
	StatusID unique identifier of status objects	smallint	2	False	1 - 1	
	StatusName friendly name/description of status object	varchar(50)	50	True		
	StatusDescription	varchar(253)	253	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())
	TargetType	varchar(50)	50	True		

Indexes

Key	Name	Columns	Unique	File Group
	PK_Status	StatusID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateStatus	True	True	After Update

SQL Script

```

CREATE TABLE [dbo].[TargetStatus]
(
    [StatusID] [smallint] NOT NULL IDENTITY(1, 1),
    [StatusName] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL,
    [StatusDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
    [ModifiedDate] [datetime] NULL,
    [CreatedDate] [datetime] NULL CONSTRAINT [DF_Status_CreatedDate] DEFAULT (getdate()),
    [TargetType] [varchar] (50) COLLATE SQL_Latin1_General_CI_AS NULL
) ON [UData]
GO
CREATE trigger [dbo].[trUpdateStatus] on [dbo].[TargetStatus] AFTER UPDATE
as
begin
    if @@rowcount = 0
        return
    if not update(ModifiedDate) update TargetStatus set ModifiedDate = getdate()
    where StatusID in (select StatusID from inserted)

    end
GO
ALTER TABLE [dbo].[TargetStatus] ADD CONSTRAINT [PK_Status] PRIMARY KEY CLUSTERED
([StatusID]) ON [UData]
GO
EXEC sp_adddextendedproperty N'MS_Description', N'collection of status objects',
    'SCHEMA', N'dbo', 'TABLE', N'TargetStatus', NULL, NULL
GO
EXEC sp_adddextendedproperty N'MS_Description', N'unique identifier of status objects',
    'SCHEMA', N'dbo', 'TABLE', N'TargetStatus', 'COLUMN', N>StatusID'
GO
EXEC sp_adddextendedproperty N'MS_Description', N'friendly name/description of status
object', 'SCHEMA', N'dbo', 'TABLE', N'TargetStatus', 'COLUMN', N>StatusName'
GO

```

Used By

[dbo].[BloodTestResults]
 [dbo].[Person]
 [dbo].[usp_SIChildStatus]
 [dbo].[usp_SIStatus]

[dbo].[TravelNotes]

MS_Description

Collection of family and travel notes

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	12
Created	3:31:08 PM Thursday, March 19, 2015
Last Modified	3:36:55 PM Thursday, March 19, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
PK_C	TravelNotesID	int	4	False	1 - 1	
FK_D	FamilyID	int	4	False		
	CreatedDate date the notes were added	datetime	8	True		(getdate())
	Notes	varchar(3000)	3000	False		
	StartDate	date	3	True		
	EndDate	date	3	True		

Indexes

Key	Name	Columns	Unique	File Group
PK_C	PK_TravelNotes	TravelNotesID	True	UData

Foreign Keys

Name	Columns
FK_TravelNotes_Family	FamilyID->[dbo].[Family].[FamilyID]

SQL Script

```
CREATE TABLE [dbo].[TravelNotes]
(
[TravelNotesID] [int] NOT NULL IDENTITY(1, 1),
[FamilyID] [int] NOT NULL,
[CreatedDate] [datetime] NULL CONSTRAINT [DF_TravelNotes_CreatedDate] DEFAULT
(getdate()),
[Notes] [varchar] (3000) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
[StartDate] [date] NULL,
[EndDate] [date] NULL
) ON [UData]
GO
ALTER TABLE [dbo].[TravelNotes] ADD CONSTRAINT [PK_TravelNotes] PRIMARY KEY CLUSTERED
([TravelNotesID]) ON [UData]
GO
ALTER TABLE [dbo].[TravelNotes] ADD CONSTRAINT [FK_TravelNotes_Family] FOREIGN KEY
([FamilyID]) REFERENCES [dbo].[Family] ([FamilyID])
GO
EXEC sp_addextendedproperty N'MS_Description', N'Collection of family and travel
notes', 'SCHEMA', N'dbo', 'TABLE', N'TravelNotes', NULL, NULL
GO
EXEC sp_addextendedproperty N'MS_Description', N'date the notes were added', 'SCHEMA',
N'dbo', 'TABLE', N'TravelNotes', 'COLUMN', N'CreatedDate'
GO
```

Uses

[dbo].[Family]

Used By

[dbo].[usp_InsertTravelNotes]

[dbo].[Units]	

Properties

Property	Value
Collation	SQL_Latin1_General_CI_AS
File Group	UData
Row Count (~)	7
Created	7:49:31 PM Friday, August 29, 2014
Last Modified	9:16:43 PM Thursday, April 9, 2015

Columns

Key	Name	Data Type	Max Length (Bytes)	Allow Nulls	Identity	Default
C	UnitsID	smallint	2	False	1 - 1	
	Units	varchar(20)	20	False		
	UnitsDescription	varchar(253)	253	True		
	ModifiedDate	datetime	8	True		
	CreatedDate	datetime	8	True		(getdate())
	HistoricUnitsCode	char(1)	1	True		

Indexes

Key	Name	Columns	Unique	File Group
C	PK_Units	UnitsID	True	UData

Triggers

Name	ANSI Nulls On	Quoted Identifier On	On
trUpdateUnits	True	True	After Update

SQL Script

```
CREATE TABLE [dbo].[Units]
(
[UnitsID] [smallint] NOT NULL IDENTITY(1, 1),
[Units] [varchar] (20) COLLATE SQL_Latin1_General_CI_AS NOT NULL,
[UnitsDescription] [varchar] (253) COLLATE SQL_Latin1_General_CI_AS NULL,
[ModifiedDate] [datetime] NULL,
```

```
[CreatedDate] [datetime] NULL CONSTRAINT [DF_Units_CreatedDate] DEFAULT (getdate()),  
[HistoricUnitsCode] [char] (1) COLLATE SQL_Latin1_General_CI_AS NULL  
) ON [UData]  
GO  
create trigger [dbo].[trUpdateUnits] on [dbo].[Units] AFTER UPDATE  
as  
begin  
if @@rowcount = 0  
return  
if not update(ModifiedDate) update Units set ModifiedDate = getdate() where Units-  
ID in (select UnitsID from inserted)  
  
end  
GO  
ALTER TABLE [dbo].[Units] ADD CONSTRAINT [PK_Units] PRIMARY KEY CLUSTERED ([UnitsID])  
ON [UData]  
GO
```

Used By

[dbo].[MediumSampleResults]
[dbo].[usp_InsertMediumSampleResults]

 Views
--

Objects

Name
dbo.vAdults
dbo.vMostRecentBloodTestResults
dbo.vMostRecentQuestionnaires
dbo.vNursingInfants
dbo.vNursingMothers
dbo.vPregnant

[dbo].[vAdults]	

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Created	8:02:42 PM Friday, May 29, 2015
Last Modified	6:50:45 PM Thursday, June 4, 2015

Columns

Name
PersonID
LastName
FirstName
Age
Gender
BirthDate
Pregnant
NursingMother

SQL Script

```
CREATE VIEW [dbo].[vAdults]
AS
SELECT      P.PersonID, P.LastName, P.FirstName, P.Age, P.Gender, P.BirthDate,
P.Pregnant, P.NursingMother
FROM        dbo.Person AS P
WHERE       (P.Age > 17) and P.isClient = 1 AND P.Pregnant = 0 AND P.NursingMother = 0
```

GO

Uses

[dbo].[Person]

[dbo].[vMostRecentBloodTestResults]	
-------------------------------------	--

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Created	5:50:00 PM Friday, May 29, 2015
Last Modified	5:26:41 PM Thursday, June 11, 2015

Columns

Name
LastName
FirstName
PersonID
BloodTestResultsID
isBaseline
SampleDate
LabSubmissionDate
LeadValue
LeadValueCategoryID
HemoglobinValue
HemoglobinValueCategoryID
HematocritValueCategoryID
LabID
BloodTestCosts
SampleTypeID
TakenAfterPropertyRemediationCompleted
ModifiedDate
CreatedDate
HematocritValue
ExcludeResult
ClientStatusID
HistoricBloodTestResultsID
HistoricLabResultsID

SQL Script

```
CREATE View [dbo].[vMostRecentBloodTestResults]
AS

Select [P].[LastName],[P].[FirstName],[P].[PersonID],[BTR].[BloodTestResultsID]
,[BTR].[isBaseline]
,[BTR].[SampleDate]
,[BTR].[LabSubmissionDate]
,[BTR].[LeadValue]
,[BTR].[LeadValueCategoryID]
,[BTR].[HemoglobinValue]
,[BTR].[HemoglobinValueCategoryID]
,[BTR].[HematocritValueCategoryID]
,[BTR].[LabID]
,[BTR].[BloodTestCosts]
,[BTR].[SampleTypeID]
,[BTR].[TakenAfterPropertyRemediationCompleted]
,[BTR].[ModifiedDate]
,[BTR].[CreatedDate]
,[BTR].[HematocritValue]
,[BTR].[ExcludeResult]
,[BTR].[ClientStatusID]
,[BTR].[HistoricBloodTestResultsID]
,[BTR].[HistoricLabResultsID] from [Person] AS [P]
JOIN [BloodTestResults] AS [BTR] on [BTR].[BloodTestResultsID] =
select top 1 [BloodTestResultsID] from [BloodTest-
Results]
where [BloodTestResults].[PersonID] = [P].[Person-
ID]
-- AND [LeadValue] > @MinLeadValue uncomment to
list most recent tests with BLL above minimum

)
WHERE [P].[isClient] = 1

GO
```

Uses

[dbo].[BloodTestResults]
[dbo].[Person]

[dbo].[vMostRecentQuestionnaires]	
--	--

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Created	5:43:01 PM Friday, May 29, 2015
Last Modified	7:41:49 AM Sunday, June 28, 2015

Columns

Name
LastName
FirstName
PersonID
QuestionnaireID
QuestionnaireDate
QuestionnaireDataSourceID
VisitRemodeledProperty
isExposedtoPeelingPaint
isTakingVitamins
NursingMother
isUsingPacifier
isUsingBottle
BitesNails
NonFood Eating
NonFood in Mouth
EatOutside
Suckling
Frequent Hand Washing
DaycareID
Created Date
Modified Date
Remodel Property Date
Remodel Property Age
Paint Date
Paint Age
Review Status ID

Author: liam

Mouthing
VisitsOldHomes
NursingInfant
Pregnant

SQL Script

```

CREATE View [dbo].[vMostRecentQuestionnaires]
AS

Select [P].[LastName], [P].[FirstName], [P].[PersonID]
    ,[Q].[QuestionnaireID]
    ,[Q].[QuestionnaireDate]
    ,[Q].[QuestionnaireDataSourceID]
    ,[Q].[VisitRemodeledProperty]
    ,[Q].[isExposedtoPeelingPaint]
    ,[Q].[isTakingVitamins]
    ,[Q].[NursingMother]
    ,[Q].[isUsingPacifier]
    ,[Q].[isUsingBottle]
    ,[Q].[BitesNails]
    ,[Q].[NonFoodEating]
    ,[Q].[NonFoodinMouth]
    ,[Q].[EatOutside]
    ,[Q].[Suckling]
    ,[Q].[FrequentHandWashing]
    ,[Q].[DaycareID]
    ,[Q].[CreatedDate]
    ,[Q].[ModifiedDate]
    ,[Q].[RemodelPropertyDate]
    ,[Q].[RemodelPropertyAge]
    ,[Q].[PaintDate]
    ,[Q].[PaintAge]
    ,[Q].[ReviewStatusID]
    ,[Q].[Mouthing]
    ,[Q].[VisitsOldHomes]
    ,[Q].[NursingInfant]
    ,[Q].[Pregnant]
    from [Person] AS [P]
JOIN [Questionnaire] AS [Q] on [Q].[QuestionnaireID] =
                                select top 1 [QuestionnaireID] from [Questionnaire]
                                where [Questionnaire].[PersonID] = [P].[PersonID]
                                -- AND [LeadValue] > @MinLeadValue uncomment to
                                list most recent tests with BLL above minimum
)
WHERE [P].[isClient] = 1

```

```
GO
```

Uses

[dbo].[Person]
[dbo].[Questionnaire]

[dbo].[vNursingInfants]	

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Created	6:38:16 PM Friday, May 29, 2015
Last Modified	8:31:45 PM Friday, May 29, 2015

Columns

Name
PersonID
LastName
FirstName
Age
Gender
NursingInfant

SQL Script

```
CREATE VIEW [dbo].[vNursingInfants]
AS
SELECT      P.PersonID, P.LastName, P.FirstName, P.Age, P.Gender,P.NursingInfant
FROM        dbo.Person AS P
WHERE       (P.NursingInfant = 1)
AND [P].[isClient] = 1

GO
```

Uses

[dbo].[Person]

[dbo].[vNursingMothers]	

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Created	6:37:20 PM Friday, May 29, 2015
Last Modified	8:35:27 PM Friday, May 29, 2015

Columns

Name
PersonID
LastName
FirstName
Age
Gender
NursingMother

SQL Script

```
CREATE VIEW [dbo].[vNursingMothers]
AS
SELECT      P.PersonID, P.LastName, P.FirstName, P.Age, P.Gender,P.NursingMother
FROM        dbo.Person AS P
WHERE       (P.NursingMother = 1)
           AND P.isClient = 1

GO
```

Uses

[dbo].[Person]

[dbo].[vPregnant]	

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Created	6:38:50 PM Friday, May 29, 2015
Last Modified	8:36:03 PM Friday, May 29, 2015

Columns

Name
PersonID
LastName
FirstName
Age
Gender
Pregnant

SQL Script

```
CREATE VIEW [dbo].[vPregnant]
AS
SELECT      P.PersonID, P.LastName, P.FirstName, P.Age, P.Gender,P.Pregnant
FROM        dbo.Person AS P
WHERE       (P.Pregnant = 1)
           AND P.isClient = 1

GO
```

Uses

[dbo].[Person]

Stored Procedures

Objects

Name
dbo.DELETE_usp_InsertPersontoStatus
dbo.DELETE_usp_SICountClients
dbo.TransProc
dbo.usp_InsertAccessAgreement
dbo.usp_InsertAccessPurpose
dbo.usp_InsertArea
dbo.usp_InsertBloodTestResults
dbo.usp_InsertBloodTestResultsNotes
dbo.usp_InsertCleanupStatus
dbo.usp_InsertConstructionType
dbo.usp_InsertContractor
dbo.usp_InsertContractortoProperty
dbo.usp_InsertContractortoRemediation
dbo.usp_InsertContractortoRemediationActionPlan
dbo.usp_InsertCountry
dbo.usp_InsertDaycare
dbo.usp_InsertDaycarePrimaryContact
dbo.usp_InsertDaycaretoProperty
dbo.usp_InsertEmployer
dbo.usp_InsertEmployertoProperty
dbo.usp_InsertEnvironmentalInvestigation
dbo.usp_InsertEthnicity
dbo.usp_InsertFamily
dbo.usp_InsertFamilyNotes
dbo.usp_InsertFamilytoPhoneNumber
dbo.usp_InsertFamilytoProperty
dbo.usp_InsertForeignFood
dbo.usp_InsertForeignFoodtoCountry
dbo.usp_InsertGiftCard
dbo.usp_InsertHobby
dbo.usp_InsertHomeRemedies
dbo.usp_InsertHouseholdSourcesofLead
dbo.usp_InsertInsuranceProvider
dbo.usp_InsertLab

dbo.usp_InsertLabNotes
dbo.usp_InsertLanguage
dbo.usp_InsertMedium
dbo.usp_InsertMediumSampleResults
dbo.usp_InsertMediumSampleResultsNotes
dbo.usp_InsertNewBloodLeadTestResultsWebScreen
dbo.usp_InsertNewClientWebScreen
dbo.usp_InsertNewFamilyWebScreen
dbo.usp_InsertNewQuestionnaireWebScreen
dbo.usp_InsertOccupation
dbo.usp_InsertPerson
dbo.usp_InsertPersonHobbyNotes
dbo.usp_InsertPersonNotes
dbo.usp_InsertPersonReleaseNotes
dbo.usp_InsertPersontoAccessAgreement
dbo.usp_InsertPersontoDaycare
dbo.usp_InsertPersontoEmployer
dbo.usp_InsertPersontoEthnicity
dbo.usp_InsertPersontoFamily
dbo.usp_InsertPersontoForeignFood
dbo.usp_InsertPersontoHobby
dbo.usp_InsertPersontoHomeRemedy
dbo.usp_InsertPersontoInsurance
dbo.usp_InsertPersontoLanguage
dbo.usp_InsertPersontoOccupation
dbo.usp_InsertPersontoPerson
dbo.usp_InsertPersontoPhoneNumber
dbo.usp_InsertPersontoProperty
dbo.usp_InsertPersontoTravelCountry
dbo.usp_InsertPersonTravelNotes
dbo.usp_InsertPhoneNumber
dbo.usp_InsertPhoneNumberType
dbo.usp_InsertProperty
dbo.usp_InsertPropertyNotes
dbo.usp_InsertPropertySampleResults
dbo.usp_InsertPropertySampleResultsNotes
dbo.usp_InsertPropertytoCleanupStatus
dbo.usp_InsertPropertytoHouseholdSourcesofLead
dbo.usp_InsertPropertytoMedium
dbo.usp_InsertQuestionnaire
dbo.usp_InsertQuestionnaireNotes

dbo.usp_InsertRemediation
dbo.usp_InsertRemediationActionPlan
dbo.usp_InsertRemediationNotes
dbo.usp_InsertSampleLevelCategory
dbo.usp_InsertSampleType
dbo.usp_InsertStatus
dbo.usp_InsertTravelNotes
dbo.usp_SLAIIBloodTestResults
dbo.usp_SLAIIBloodTestResults2
dbo.usp_SLAIIBloodTestResultsMetaData
dbo.usp_SIChildStatus
dbo.usp_SIClientFollowUp
dbo.usp_SIColumnDetails
dbo.usp_SICountAdults
dbo.usp_SICountBloodLeadLevels
dbo.usp_SICountBloodTests
dbo.usp_SICountClients
dbo.usp_SICountFamilyMembers
dbo.usp_SICountHomeVisitSoilSample
dbo.usp_SICountNewClients
dbo.usp_SICountNewPeople
dbo.usp_SICountNursingInfants
dbo.usp_SICountNursingMothers
dbo.usp_SICountPeople
dbo.usp_SICountPeopleByAge
dbo.usp_SICountPeopleByAgeGroup
dbo.usp_SICountPeopleByLastName
dbo.usp_SICountPregnantWomen
dbo.usp_SIDaycare
dbo.usp_SIEditBloodTestResultsWebScreenInformation
dbo.usp_SIEditClientInfoWebScreenInformation
dbo.usp_SIEditFamilyWebScreenInformation
dbo.usp_SIEditPropertyWebScreenInformation
dbo.usp_SIEditQuestionnaireWebScreenInformation
dbo.usp_SIFamilyMembers
dbo.usp_SIFamilyNameToProperty
dbo.usp_SIHobby
dbo.usp_SIInsertedData
dbo.usp_SIInsertedDataSimplified
dbo.usp_SILabName
dbo.usp_SLLListAllFamilyMembers

dbo.usp_SIListClientsByCreatedate
dbo.usp_SIListClientsByModifieddate
dbo.usp_SIListFamilies
dbo.usp_SIListFamilyMembers
dbo.usp_SIListNursingWomenbyCreateDateRange
dbo.usp_SIListPeoplebyCreateDateRange
dbo.usp_SLLListPotentialDuplicatePeople
dbo.usp_SLLListPotentialDuplicateProperties
dbo.usp_SIListPregnantWomenbyCreateDateRange
dbo.usp_SLMostRecentBloodTestResults
dbo.usp_SIPersonNotes
dbo.usp_SIPersonsToEthnicity
dbo.usp_SIPersonsToLanguage
dbo.usp_SIRelationshipTypes
dbo.usp_SIStatus
dbo.usp_SISummaryReport
dbo.usp_SISummaryReport_MetaData
dbo.usp_SITargetSampleType
dbo.usp_upBloodTestResults
dbo.usp_upBloodTestResultsWebScreen
dbo.usp_upClientFlag
dbo.usp_upClientWebScreen
dbo.usp_upFamily
dbo.usp_upFamilytoProperty
dbo.usp_upFamilyWebScreen
dbo.usp_upOccupation
dbo.usp_upPerson
dbo.usp_upProperty
dbo.usp_upQuestionnaire
dbo.usp_upQuestionnaireWebScreen
dbo.usp.LogError
dbo.uspPrintError

[dbo].[DELETE_usp_InsertPersontoStatus]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@StatusID	int	4
@StatusDate	date	3

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PersontoStatus records
-- =====

CREATE PROCEDURE [dbo].[DELETE_usp_InsertPersontoStatus]    -- usp_InsertPersontoStatus
    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @StatusID int = NULL,
    @StatusDate date = NULL

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into PersontoStatus( PersonID, StatusID, StatusDate )
        Values ( @PersonID, @StatusID, @StatusDate )
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[DELETE_usp_SICountClients]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Max_Age	int	4
@Start_Date	datetime	8
@End_Date	datetime	8
@DEBUG	bit	1

SQL Script

```
-- =====

-- Author:      Liam Thier

-- Create date: 20150604

-- Description:    procedure returns the number of

--                  entries in the persons table where

--                  isClient = 1 filtered by age

--                  and Report Dates

-- =====

CREATE PROCEDURE [dbo].[DELETE_usp_SICountClients]
    -- Add the parameters for the stored procedure here

    @Max_Age int = NULL,
    @Start_Date datetime = '18000101',
    @End_Date datetime = NULL,
    @DEBUG bit = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

IF (@End_Date IS NULL)
    SELECT @End_Date = GetDate();

DECLARE @spexecutesqlStr NVARCHAR(4000), @Recompile BIT = 1, @ErrorLogID int, @Max-
Age int;

BEGIN TRY
    SELECT @spexecutesqlStr = 'SELECT TotalClients = count([PersonId]) from
[person] WHERE isClient = 1'

    IF (@Max_Age IS NOT NULL)
        BEGIN
            SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND [Age] <= @MaxAge';
        END

    IF (@Start_Date IS NOT NULL)
        SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND      ( CreatedDate >=
@StartDate OR ModifiedDate >= @StartDate ) '

    IF @Recompile = 1
        SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

    EXEC [sp_executesql] @spexecutesqlStr
    , N'@MaxAge VARCHAR(50), @StartDate datetime'
    , @MaxAge = @Max_Age
    , @StartDate = @Start_Date

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[TransProc]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	False

Parameters

Name	Data Type	Max Length (Bytes)
@PriKey	int	4
@CharCol	char(3)	3

SQL Script

```
SET QUOTED_IDENTIFIER OFF
GO
CREATE PROCEDURE [dbo].[TransProc] @PriKey INT, @CharCol CHAR(3) AS
BEGIN TRANSACTION InProc
INSERT INTO TestTrans VALUES (@PriKey, @CharCol)
INSERT INTO TestTrans VALUES (@PriKey + 1, @CharCol)
COMMIT TRANSACTION InProc;
GO
```

[dbo].[usp_InsertAccessAgreement]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@AccessPurposeID	int	4	
@Notes	varchar(3000)	3000	
@AccessAgreementFile	varbinary(max)	max	
@PropertyID	int	4	
@InsertedAccessAgreementID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new
--               AccessAgreement records
-- =====
-- HISTORY
-- 12/13/2014    modified procedure to accept OUTPUT parameters

CREATE PROCEDURE [dbo].[usp_InsertAccessAgreement]      -- usp_InsertAccessAgreement
-- Add the parameters for the stored procedure here
@AccessPurposeID int = NULL,
@Notes varchar(3000) = NULL,
@AccessAgreementFile varbinary(max) = NULL,
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_InsertAccessAgreement

```
@PropertyID int = NULL,
@InsertedAccessAgreementID int OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
    -- Insert statements for procedure here

    BEGIN TRY
        INSERT into AccessAgreement (AccessPurposeID, AccessAgreementFile, PropertyID)
            Values ( @AccessPurposeID, @AccessAgreementFile, @PropertyID);
        SELECT @InsertedAccessAgreementID = SCOPE_IDENTITY();

        IF (@NOTES IS NOT NULL)
            BEGIN TRY
                INSERT into AccessAgreementNotes (AccessAgreementID, Notes)
                    Values (@InsertedAccessAgreementID, @Notes)
            END TRY
            BEGIN CATCH
                -- Call procedure to print error information.

                EXECUTE dbo.uspPrintError;

                -- Roll back any active or uncommittable transactions before
                -- inserting information in the ErrorLog.

                IF XACT_STATE() <> 0
                    BEGIN
                        ROLLBACK TRANSACTION;
                    END

                EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
            END CATCH;
        END TRY
        BEGIN CATCH
            -- Call procedure to print error information.

            EXECUTE dbo.uspPrintError;

            -- Roll back any active or uncommittable transactions before
            -- inserting information in the ErrorLog.

            IF XACT_STATE() <> 0
                BEGIN
                    ROLLBACK TRANSACTION;
                END
        END CATCH;
    END TRY
END
```

Author: liam

```
EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[AccessAgreement]
[dbo].[AccessAgreementNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertAccessPurpose]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@AccessPurposeName	varchar(50)	50	
@AccessPurposeDescription	varchar(250)	250	
@AccessPurposeID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new AccessPurpose records
-- =====
-- HISTORY
-- 12/13/2014    modified procedure to accept OUTPUT parameters

CREATE PROCEDURE [dbo].[usp_InsertAccessPurpose]      -- usp_InsertAccessPurpose
    -- Add the parameters for the stored procedure here
    @AccessPurposeName varchar(50) = NULL,
    @AccessPurposeDescription varchar(250) = NULL,
    @AccessPurposeID int OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into AccessPurpose ( AccessPurposeName, AccessPurposeDescription)
        Values ( @AccessPurposeName, @AccessPurposeDescription);
    SELECT @AccessPurposeID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[AccessPurpose]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertArea]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@AreaDescription	varchar(250)	250	
@AreaName	varchar(50)	50	
@NewAreaID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new Area records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertArea] -- usp_InsertArea
    -- Add the parameters for the stored procedure here
    @AreaDescription varchar(250) = NULL,
    @AreaName varchar(50) = NULL,
    @NewAreaID int OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
```

```
-- Insert statements for procedure here

BEGIN TRY
    INSERT into Area ( AreaDescription, HistoricAreaID)
        Values ( @AreaDescription, @AreaName);
    SELECT @NewAreaID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Area]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertBloodTestResults]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@isBaseline	bit	1	
@PersonID	int	4	
@SampleDate	date	3	
@LabSubmissionDate	date	3	
@LeadValue	numeric(4,1)	5	
@LeadValueCategoryID	tinyint	1	
@HemoglobinValue	numeric(4,1)	5	
@HemoglobinValueCategoryID	tinyint	1	
@HematocritValueCategoryID	tinyint	1	
@LabID	int	4	
@ClientStatusID	smallint	2	
@BloodTestCosts	money	8	
@sampleTypeID	tinyint	1	
@New_Notes	varchar(3000)	3000	
@TakenAfterPropertyRemediationCompleted	bit	1	
@BloodTestResultID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new BloodTestResults records
```

```
-- =====

CREATE PROCEDURE [dbo].[usp_InsertBloodTestResults] -- usp_InsertBloodTestResults

    -- Add the parameters for the stored procedure here

    @isBaseline bit = NULL,
    @PersonID int = NULL,
    @SampleDate date = NULL,
    @LabSubmissionDate date = NULL,
    @LeadValue numeric(4,1) = NULL,
    @LeadValueCategoryID tinyint = NULL,
    @HemoglobinValue numeric(4,1) = NULL,
    @HemoglobinValueCategoryID tinyint = NULL, -- lookup in the database

    @HematocritValueCategoryID tinyint = NULL, -- lookup in the database

    @LabID int = NULL,
    @ClientStatusID smallint = NULL,
    @BloodTestCosts money = NULL,
    @sampleTypeID tinyint = NULL,
    @New_Notes varchar(3000) = NULL,
    @TakenAfterPropertyRemediationCompleted bit = NULL,
    @BloodTestResultID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ExistsPersonID int -- does the person have a record in BloodTestResults
    table

        , @ErrorLogID int, @NotesID int;
    -- Handle Null sampleDate?

    -- Handle Null LabSubmissionDate?

    -- check if the person exists

    IF NOT EXISTS (select PersonID from Person where PersonID = @PersonID)
    BEGIN
        RAISERROR ('Person does not exist. Cannot create a BloodtestResult record', 11,
        -1);
        RETURN;
    END

    -- check if the person has a record in BloodTestResults Table
```

```
select @ExistsPersonID = PersonID from BloodTestResults

-- Insert statements for procedure here

BEGIN TRY
    -- Determine if this person already has an entry in BloodTestResults and set is-
    Baseline appropriately.

    IF ( @isBaseline is NULL ) -- nothing passed in for baseline

        BEGIN
            IF ( @ExistsPersonID is not NULL )
                BEGIN
                    SET @isBaseline = 0;
                END
            ELSE -- the person has no entry in BloodTestResults, this is a baseline
entry

                BEGIN
                    SET @isBaseline = 1;
                END
            END
        ELSE IF ( @isBaseline = 0 ) -- this should not be a baseline entry according to
passed in argument

            BEGIN
                IF (@ExistsPersonID is NULL) -- the person does not have an entry in Blood-
                TestResults, this is a baseline entry

                    BEGIN
                        Set @isBaseline = 1;
                    END
                END
            ELSE IF ( @isBaseline = 1 ) -- this should be a baseline entry according to
passed in argument

                BEGIN
                    IF (@ExistsPersonID is not NULL) -- the person already has an entry in
BloodTestResults, this isn't a baseline entry

                        BEGIN
                            Set @isBaseline = 0;
                        END
                    END
                END

            INSERT into BloodTestResults ( isBaseline, PersonID, SampleDate, LabSubmission-
Date, LeadValue, LeadValueCategoryID,
                                         HemoglobinValue, HemoglobinValueCategoryID,
                                         HematocritValueCategoryID, LabID, ClientStatusID,
                                         BloodTestCosts, SampleTypeID, TakenAfter-
PropertyRemediationCompleted)
                Values ( @isBaseline, @PersonID, @SampleDate, @LabSubmissionDate,
@LeadValue, @LeadValueCategoryID,
                                         @HemoglobinValue, @HemoglobinValueCategoryID, @Hematocrit-
ValueCategoryID, @LabID, @ClientStatusID,
```

```
    @BloodTestCosts, @SampleTypeID, @TakenAfterProperty-
    RemediationCompleted);
    SELECT @BloodTestResultID = SCOPE_IDENTITY();

    IF (@New_Notes IS NOT NULL)
        EXEC [dbo].[usp_InsertBloodTestResultsNotes]
            @BloodtestResults_ID = @BloodTestResultID,
            @Notes = @New_Notes,
            @InsertedNotesID = @NotesID OUTPUT
    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
            BEGIN
                ROLLBACK TRANSACTION;
            END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END
```

GO

Uses

[dbo].[BloodTestResults]
[dbo].[Person]
[dbo].[usp_InsertBloodTestResultsNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertNewBloodLeadTestResultsWebScreen]

[dbo].[usp_InsertBloodTestResultsNotes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@BloodTestResults_ID	int	4	
@Notes	varchar(3000)	3000	
@InsertedNotesID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150215
-- Description: stored procedure to insert BloodTestResults notes
-- =====

CREATE PROCEDURE [dbo].[usp_InsertBloodTestResultsNotes]
    -- Add the parameters for the stored procedure here
    @BloodTestResults_ID int = NULL,
    @Notes VARCHAR(3000) = NULL,
    @InsertedNotesID INT OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @ErrorLogID int

    -- Insert statements for procedure here
    BEGIN TRY -- update BloodTestResults information
```

```
INSERT INTO BloodTestResultsNotes (BloodTestResultsID, Notes)
    values (@BloodTestResults_ID, @Notes);
SET @InsertedNotesID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    -- inserting information in the ErrorLog.

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[BloodTestResultsNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertBloodTestResults]
[dbo].[usp_upBloodTestResults]

[dbo].[usp_InsertCleanupStatus]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@CleanupStatusDescription	varchar(200)	200	
@CleanupStatusName	varchar(25)	25	
@NewCleanupStatusID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new CleanupStatus records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertCleanupStatus] -- usp_InsertCleanupStatus
    -- Add the parameters for the stored procedure here
    @CleanupStatusDescription varchar(200) = NULL,
    @CleanupStatusName varchar(25) = NULL,
    @NewCleanupStatusID int OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into CleanupStatus ( CleanupStatusDescription, CleanupStatusName)
        Values ( @CleanupStatusDescription, @CleanupStatusName);
    SELECT @NewCleanupStatusID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[CleanupStatus]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertConstructionType]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@ConstructionTypeDescription	varchar(250)	250	
@ConstructionTypeName	varchar(50)	50	
@NewConstructionTypeID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new ConstructionType records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertConstructionType]    -- usp_InsertConstructionType
    -- Add the parameters for the stored procedure here
    @ConstructionTypeDescription varchar(250) = NULL,
    @ConstructionTypeName varchar(50) = NULL,
    @NewConstructionTypeID int OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into ConstructionType ( ConstructionTypeDescription, ConstructionTypeName)
        Values ( @ConstructionTypeDescription, @ConstructionTypeName);
    SELECT @NewConstructionTypeID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[ConstructionType]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertContractor]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@ContractorDescription	varchar(250)	250	
@ContractorName	varchar(50)	50	
@NewContractorID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new Contractor records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertContractor]    -- usp_InsertContractor
    -- Add the parameters for the stored procedure here
    @ContractorDescription varchar(250) = NULL,
    @ContractorName varchar(50) = NULL,
    @NewContractorID int OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into Contractor ( ContractorDescription, ContractorName)
        Values ( @ContractorDescription, @ContractorName);
    SELECT @NewContractorID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
```

GO

Uses

[dbo].[Contractor]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertContractortoProperty]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ContractorID	int	4
@PropertyID	int	4
@StartDate	date	3
@EndDate	date	3

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new ContractortoProperty records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertContractortoProperty]    -- usp_InsertContractortoProperty

    -- Add the parameters for the stored procedure here

    @ContractorID int = NULL,
    @PropertyID int = NULL,
    @StartDate date = NULL,
    @EndDate date = NULL

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into ContractortoProperty ( ContractorID, PropertyID, StartDate, End-
Date)
        Values ( @ContractorID, @PropertyID, @StartDate, @EndDate);

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[ContractortoProperty]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertContractortoRemediation]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ContractorID	int	4
@RemediationID	int	4
@StartDate	date	3
@EndDate	date	3
@isSubContractor	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new ContractortoRemediation records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertContractortoRemediation] -- usp_InsertContractortoRemediation
-- Add the parameters for the stored procedure here

@ContractorID int = NULL,
@RemediationID int = NULL,
@StartDate date = NULL,
@EndDate date = NULL,
@isSubContractor bit = NULL

AS
```

Author: liam

```
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
    -- Insert statements for procedure here

    BEGIN TRY
        INSERT into ContractortoRemediation ( ContractorID, RemediationID, StartDate,
EndDate, isSubContractor)
            Values ( @ContractorID, @RemediationID, @StartDate, @EndDate, @is-
SubContractor);
        SELECT SCOPE_IDENTITY();
    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END
```

GO

Uses

[dbo].[ContractortoRemediation]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertContractortoRemediationActionPlan]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ContractorID	int	4
@RemediationActionPlanID	int	4
@StartDate	date	3
@EndDate	date	3
@isSubContractor	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new ContractortoRemediationPlan records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertContractortoRemediationActionPlan] -- usp_Insert-
ContractortoRemediationPlan

    -- Add the parameters for the stored procedure here

    @ContractorID int = NULL,
    @RemediationActionPlanID int = NULL,
    @StartDate date = NULL,
    @EndDate date = NULL,
    @isSubContractor bit = NULL

AS
```

Author: liam

```
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
    -- Insert statements for procedure here

    BEGIN TRY
        INSERT into ContractortoRemediationActionPlan ( ContractorID, Remediation-
ActionPlanID, StartDate, EndDate, isSubContractor)
            Values ( @ContractorID, @RemediationActionPlanID, @StartDate, @End-
Date, @isSubContractor);
    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END
GO
```

Uses

[dbo].[ContractortoRemediationActionPlan]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertCountry]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@CountryName	varchar(50)	50	
@NewCountryID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new Country records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertCountry]    -- usp_InsertCountry
    -- Add the parameters for the stored procedure here
    @CountryName varchar(50) = NULL,
    @NewCountryID int OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
```

Author: liam

```
-- Insert statements for procedure here

BEGIN TRY
    INSERT into Country ( CountryName)
        Values ( @CountryName);
    SELECT @NewCountryID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Country]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertDaycare]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@DaycareName	varchar(50)	50	
@DaycareDescription	varchar(200)	200	
@newDayCareID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new Daycare records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertDaycare]    -- usp_InsertDaycare
    -- Add the parameters for the stored procedure here

    @DaycareName varchar(50) = NULL,
    @DaycareDescription varchar(200) = NULL,
    @newDayCareID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into Daycare ( DaycareName, DaycareDescription )
        Values ( @DaycareName, @DaycareDescription );
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Daycare]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertDaycarePrimaryContact]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@DaycareID	int	4
@PersonID	int	4
@ContactPriority	tinyint	1
@PrimaryPhoneNumberID	int	4

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new DaycarePrimaryContact records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertDaycarePrimaryContact]      -- usp_InsertDaycarePrimary-
Contact

    -- Add the parameters for the stored procedure here

    @DaycareID int = NULL,
    @PersonID int = NULL,
    @ContactPriority tinyint = NULL,
    @PrimaryPhoneNumberID int = NULL

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into DaycarePrimaryContact ( DayCareID, PersonID, ContactPriority,
PrimaryPhoneNumberID )
        Values ( @DayCareID, @PersonID, @ContactPriority, @PrimaryPhone-
NumberID );
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[DaycarePrimaryContact]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertDaycaretoProperty]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@DaycareID	int	4
@PropertyID	int	4
@StartDate	date	3
@EndDate	date	3

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new DaycaretoProperty records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertDaycaretoProperty]    -- usp_InsertDaycaretoProperty
    -- Add the parameters for the stored procedure here
    @DaycareID int = NULL,
    @PropertyID int = NULL,
    @StartDate date = NULL,
    @EndDate date = NULL

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into DaycaretoProperty ( DaycareID, PropertyID, StartDate, EndDate)
        Values ( @DaycareID, @PropertyID, @StartDate, @EndDate);

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[DaycaretoProperty]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertEmployer]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@EmployerName	varchar(50)	50	
@NewEmployerID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new Employer records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertEmployer]    -- usp_InsertEmployer
    -- Add the parameters for the stored procedure here
    @EmployerName VARCHAR(50) = NULL,
    @NewEmployerID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
```

Author: liam

```
-- Insert statements for procedure here

BEGIN TRY
    INSERT into Employer ( EmployerName )
        Values ( @EmployerName );
    SELECT @NewEmployerID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Employer]
[dbo].[usp.LogError]
[dbo].[uspPrintError]



[dbo].[usp_InsertEmployertoProperty]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@EmployerID	int	4
@PropertyID	int	4
@StartDate	date	3
@EndDate	date	3

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new EmployertoProperty records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertEmployertoProperty]      -- usp_InsertEmployertoProperty
    -- Add the parameters for the stored procedure here
    @EmployerID int = NULL,
    @PropertyID int = NULL,
    @StartDate date = NULL,
    @EndDate date = NULL
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into EmployertoProperty ( EmployerID, PropertyID, StartDate, EndDate)
        Values ( @EmployerID, @PropertyID, @StartDate, @EndDate);
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[EmployertoProperty]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertEnvironmentalInvestigation]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@ConductEnvironmentalInvestigation	bit	1	
@ConductEnvironmentalInvestigationDecisionDate	date	3	
@Cost	money	8	
@EnvironmentalInvestigationDate	date	3	
@PropertyID	int	4	
@StartDate	date	3	
@EndDate	date	3	
@NewEnvironmentalInvestigation	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new EnvironmentalInvestigation records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertEnvironmentalInvestigation] -- usp_Insert-
EnvironmentalInvestigation

    -- Add the parameters for the stored procedure here

    @ConductEnvironmentalInvestigation bit = NULL,
    @ConductEnvironmentalInvestigationDecisionDate date = NULL,
    @Cost money = NULL,
```

```
@EnvironmentalInvestigationDate date = NULL,
@PropertyID int = NULL,
@StartDate date = NULL,
@EndDate date = NULL,
@NewEnvironmentalInvestigation int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
    -- Insert statements for procedure here

    BEGIN TRY
        INSERT into EnvironmentalInvestigation ( ConductEnvironmentalInvestigation,
ConductEnvironmentalInvestigationDecisionDate,
                                                Cost, EnvironmentalInvestigationDate,
PropertyID, StartDate, EndDate )
            Values ( @ConductEnvironmentalInvestigation, @ConductEnvironmental-
InvestigationDecisionDate,
                    @Cost, @EnvironmentalInvestigationDate, @PropertyID,
@StartDate, @EndDate );
        SELECT @NewEnvironmentalInvestigation = SCOPE_IDENTITY();
    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END
GO
```

Uses

[dbo].[EnvironmentalInvestigation]

[dbo].[usp.LogError]

Author: liam

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_InsertEnvironmentalInvestigation

[dbo].[uspPrintError]

Author: liam

[dbo].[usp_InsertEthnicity]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@Ethnicity	varchar(50)	50	
@NewEthnicityID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new Ethnicity records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertEthnicity] -- usp_InsertEthnicity
    -- Add the parameters for the stored procedure here
    @Ethnicity varchar(50) = NULL,
    @NewEthnicityID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
```

Author: liam

```
-- Insert statements for procedure here

BEGIN TRY
    INSERT into Ethnicity ( Ethnicity )
        Values ( @Ethnicity );
    SELECT @NewEthnicityID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Ethnicity]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertFamily]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@LastName	varchar(50)	50	
@NumberofSmokers	tinyint	1	
@PrimaryLanguageID	tinyint	1	
@Notes	varchar(3000)	3000	
@ForeignTravel	bit	1	
@New_Travel_Notes	varchar(3000)	3000	
@Travel_Start_Date	date	3	
@Travel_End_Date	date	3	
@Pets	tinyint	1	
@Petsonandout	bit	1	
@PrimaryPropertyID	int	4	
@FrequentlyWashPets	bit	1	
@FID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140205
-- Description: Stored Procedure to insert new Family information
-- =====

CREATE PROCEDURE [dbo].[usp_InsertFamily]
    -- Add the parameters for the stored procedure here
```

```

@LastName varchar(50) = NULL,
@NumberofSmokers tinyint = 0,
@PrimaryLanguageID tinyint = 1,
@Notes varchar(3000) = NULL,
@ForeignTravel bit = NULL,
@New_Travel_Notes varchar(3000) = NULL,
@Travel_Start_Date date = NULL,
@Travel_End_Date date = NULL,
@Pets tinyint = NULL,
@Petsonandout bit = NULL,
@PrimaryPropertyID int = NULL,
@FrequentlyWashPets bit = NULL,
@FID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    -- Insert statements for procedure here

    DECLARE @ErrorLogID int, @FamilyNotesReturnValue int, @InsertedFamilyNotesID int
        , @TravelNotesReturnValue int, @InsertedTravelNotesID int;

    BEGIN TRY -- insert Family

        BEGIN TRANSACTION InsertFamilyTransaction
            INSERT into Family ( LastName, NumberofSmokers, PrimaryLanguageID, Pets,
        Petsonandout
            , PrimaryPropertyID, FrequentlyWashPets, ForeignTravel)
            Values (@LastName, @NumberofSmokers, @PrimaryLanguageID, @Pets,
        @Petsonandout
            , @PrimaryPropertyID, @FrequentlyWashPets, @ForeignTravel)
            SET @FID = SCOPE_IDENTITY(); -- uncomment to return primary key of
        inserted values

        IF (@Notes IS NOT NULL)
            EXEC      @FamilyNotesReturnValue = [dbo].[usp_InsertFamilyNotes]
                @Family_ID = @FID,
                @Notes = @Notes,
                @InsertedNotesID = @InsertedFamily-
        NotesID OUTPUT

        IF (@New_Travel_Notes IS NOT NULL)
            EXEC      @TravelNotesReturnValue = [dbo].[usp_InsertTravelNotes]
                @Family_ID = @FID,
                @Travel_Notes = @New_Travel_Notes,
                @Start_Date = @Travel_Start_Date,
                @End_Date = @Travel_End_Date,
                @InsertedNotesID = @InsertedTravelNotesID OUTPUT
    END TRY
    BEGIN CATCH
        IF (@FamilyNotesReturnValue < 0)
            SET @ErrorLogID = 1;
        IF (@TravelNotesReturnValue < 0)
            SET @ErrorLogID = 2;
        IF (@ErrorLogID > 0)
            EXECUTE sp_LogError @ErrorLogID, 'Error inserting family record';
        ROLLBACK TRANSACTION InsertFamilyTransaction;
    END CATCH
END

```

```
COMMIT TRANSACTION InsertFamilyTransaction
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER();
END CATCH;
END
GO
```

Uses

[dbo].[Family]
[dbo].[usp_InsertFamilyNotes]
[dbo].[usp_InsertTravelNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertNewFamilyWebScreen]

[dbo].[usp_InsertFamilyNotes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@Family_ID	int	4	
@Notes	varchar(3000)	3000	
@InsertedNotesID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150214
-- Description: stored procedure to insert family notes
-- =====

CREATE PROCEDURE [dbo].[usp_InsertFamilyNotes]
    -- Add the parameters for the stored procedure here

    @Family_ID int = NULL,
    @Notes VARCHAR(3000) = NULL,
    @InsertedNotesID INT OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @ErrorLogID int

    -- Insert statements for procedure here

    BEGIN TRY -- update Family information

```

```
INSERT INTO FamilyNotes (FamilyID, Notes)
    values (@Family_ID, @Notes);
SET @InsertedNotesID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    -- inserting information in the ErrorLog.

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER();
END CATCH;
END
GO
```

Uses

[dbo].[FamilyNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertFamily]
[dbo].[usp_upFamily]

[dbo].[usp_InsertFamilytoPhoneNumber]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@FamilyID	int	4
@PhoneNumberID	int	4
@NumberPriority	tinyint	1
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20150404
-- Description: Stored Procedure to insert new
--               FamilytoPhoneNumber records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertFamilytoPhoneNumber]
    -- Add the parameters for the stored procedure here

    @FamilyID int = NULL,
    @PhoneNumberID int = NULL,
    @NumberPriority tinyint = NULL,
    @DEBUG bit = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int, @ExistingNumberPriority tinyint, @FamilytoPhoneNumber-
Exists bit = 0
    , @FamilyPhonePriorityExists int = 0, @SwapPriority tinyint;
-- Insert statements for procedure here

BEGIN TRY
    -- see if family already has that number

    SELECT @FamilytoPhoneNumberExists = 1, @SwapPriority = NumberPriority from
FamilytoPhoneNumber where FamilyID = @FamilyID and PhoneNumberID = @PhoneNumberID
    -- see if the family already has a number with same priority get the PPhone-
NumberID

    SELECT @FamilyPhonePriorityExists = PhoneNumberID from FamilytoPhoneNumber
where FamilyID = @FamilyID and NumberPriority = @NumberPriority

    -- If the family is already associated with the phone number

    IF (@FamilytoPhoneNumberExists = 1)
    BEGIN
        -- if the priority is the same do nothing

        IF (@SwapPriority = @NumberPriority)
            RETURN;
        ELSE IF (@FamilyPhonePriorityExists = 0) -- there are no numbers for that
family with that priority, set the New number priority for the specified number

            update FamilytoPhoneNumber set NumberPriority = @NumberPriority where
FamilyID = @FamilyID and PhoneNumberID = @PhoneNumberID

        ELSE -- there is another number for that family with the desired priority,
swap priorities

            BEGIN
                -- Set the New number priority for the specified number

                update FamilytoPhoneNumber set NumberPriority = @NumberPriority where
FamilyID = @FamilyID and PhoneNumberID = @PhoneNumberID

                -- Set number priority for number that had the existing @NumberPriority
to the previous priority from the passed in phone number

                update FamilytoPhoneNumber set NumberPriority = @SwapPriority where
FamilyID = @FamilyID and PhoneNumberID = @FamilyPhonePriorityExists
            END
        END
        ELSE -- the family is not associated with that phone number

        BEGIN
            -- there are no numbers for that family with that priority

            IF (@FamilyPhonePriorityExists = 0)
```

```
BEGIN
    INSERT into FamilytoPhoneNumber( FamilyID, PhoneNumberID, Number-
Priority)
        Values ( @FamilyID, @PhoneNumberID, @NumberPriority )
END
ELSE
BEGIN
    -- Insert the New number and priority

    INSERT into FamilytoPhoneNumber( FamilyID, PhoneNumberID, Number-
Priority)
        Values ( @FamilyID, @PhoneNumberID, @NumberPriority )

    -- determine next priority

    select @SwapPriority = max(NumberPriority)+1 from FamilytoPhoneNumber
where FamilyID = @FamilyID
        -- Set number priority for number that had the existing @NumberPriority
        to the lowest priority

        update FamilytoPhoneNumber set NumberPriority = @SwapPriority where
FamilyID = @FamilyID and PhoneNumberID = @FamilyPhonePriorityExists
    END
END
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before

    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    THROW
    -- RETURN ERROR_NUMBER()

    END CATCH;
END

GO
```

Uses

[dbo].[FamilytoPhoneNumber]

Author: liam

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_InsertFamilyto-
PhoneNumber

[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertNewFamilyWebScreen]
[dbo].[usp_upFamilyWebScreen]

[dbo].[usp_InsertFamilytoProperty]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@FamilyID	int	4	
@PropertyID	int	4	
@PropertyLinkTypeID	int	4	
@StartDate	date	3	
@EndDate	date	3	
@isPrimaryResidence	bit	1	
@DEBUG	bit	1	
@NewFamilytoPropertyID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 201504817
-- Description: Stored Procedure to insert new FamilytoProperty records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertFamilytoProperty]    -- usp_InsertFamilytoProperty

    -- Add the parameters for the stored procedure here

    @FamilyID int = NULL,
    @PropertyID int = NULL,
    @PropertyLinkTypeID int = NULL,
    @StartDate date = NULL,
    @EndDate date = NULL,
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_InsertFamilytoProperty

```
@isPrimaryResidence bit = NULL,
@DEBUG bit = NULL,
@NewFamilytoPropertyID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int, @FamilytoPropertyID int, @update_return_value int;
    -- Insert statements for procedure here

    BEGIN TRY
        select @FamilytoPropertyID = FamilytoPropertyID from FamilytoProperty where
FamilyId = @FamilyID and PropertyID = @PropertyID

        IF @FamilytoPropertyID IS NOT NULL
        BEGIN
            EXEC @update_return_value = usp_upFamilytoProperty
                @FamilytoPropertyID = @FamilytoPropertyID,
                @PropertyLinkTypeID = @PropertyLinkTypeID,
                @StartDate = @StartDate,
                @EndDate = @EndDate,
                @isPrimaryResidence = @isPrimaryResidence,
                @DEBUG = @DEBUG
        END
        ELSE
            INSERT into FamilytoProperty( FamilyID, PropertyID, PropertyLinkTypeID,
StartDate, EndDate, isPrimaryResidence)
                Values ( @FamilyID, @PropertyID, @PropertyLinkTypeID, @Start-
Date, @EndDate, @isPrimaryResidence )
            --SELECT @NewFamilytoPropertyID = SCOPE_IDENTITY();

    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
```

Author: liam

```
END  
GO
```

Uses

[dbo].[FamilytoProperty]
[dbo].[usp_upFamilytoProperty]
[dbo].[uspLogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertNewFamilyWebScreen]
[dbo].[usp_upFamilyWebScreen]

[dbo].[usp_InsertForeignFood]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@ForeignFoodName	varchar(50)	50	
@ForeignFoodDescription	varchar(256)	256	
@NewForeignFoodID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new ForeignFood records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertForeignFood] -- usp_InsertForeignFood
    -- Add the parameters for the stored procedure here
    @ForeignFoodName varchar(50) = NULL,
    @ForeignFoodDescription varchar(256) = NULL,
    @NewForeignFoodID int OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into ForeignFood ( ForeignFoodName, ForeignFoodDescription )
        Values ( @ForeignFoodName, @ForeignFoodDescription );
    SELECT @NewForeignFoodID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER();
END CATCH;
END
GO
```

Uses

[dbo].[ForeignFood]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertForeignFoodtoCountry]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@ForeignFoodID	int	4
@CountryID	tinyint	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new ForeignFoodtoCountry records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertForeignFoodtoCountry]    -- usp_InsertForeignFoodto-
Country

    -- Add the parameters for the stored procedure here

    @ForeignFoodID int = NULL,
    @CountryID tinyint = NULL

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
```

```
DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into ForeignFoodtoCountry ( ForeignFoodID, CountryID ) --, StartDate,
EndDate)

        Values ( @ForeignFoodID, @CountryID ) -- , @StartDate, @EndDate);

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[ForeignFoodtoCountry]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertGiftCard]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@GiftCardValue	money	8	
@IssueDate	date	3	
@PersonID	int	4	
@NewGiftCardID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new GiftCard records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertGiftCard]    -- usp_InsertGiftCard
    -- Add the parameters for the stored procedure here

    @GiftCardValue money = NULL,
    @IssueDate date = NULL,
    @PersonID int = NULL,
    @NewGiftCardID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    IF EXISTS (SELECT PersonID from Person where PersonID = @PersonID) print
'Person exists'
    INSERT into GiftCard ( GiftCardValue, IssueDate, PersonID )
        Values ( @GiftCardValue, @IssueDate, @PersonID );
    SELECT @NewGiftCardID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[GiftCard]
[dbo].[Person]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertHobby]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@HobbyName	varchar(50)	50	
@HobbyDescription	varchar(256)	256	
@LeadExposure	bit	1	
@NewHobbyID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new Hobby records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertHobby] -- usp_InsertHobby
    -- Add the parameters for the stored procedure here

    @HobbyName varchar(50) = NULL,
    @HobbyDescription varchar(256) = NULL,
    @LeadExposure bit = NULL,
    @NewHobbyID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into Hobby ( HobbyName, HobbyDescription, LeadExposure )
        Values ( @HobbyName, @HobbyDescription, @LeadExposure );
    SELECT @NewHobbyID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[Hobby]
 [dbo].[usp.LogError]
 [dbo].[uspPrintError]



Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@HomeRemedyName	varchar(50)	50	
@HomeRemedyDescription	varchar(256)	256	
@NewHomeRemedyID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new HomeRemedies records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertHomeRemedies]    -- usp_InsertHomeRemedies
    -- Add the parameters for the stored procedure here
    @HomeRemedyName varchar(50) = NULL,
    @HomeRemedyDescription varchar(256) = NULL,
    @NewHomeRemedyID int OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into HomeRemedy ( HomeRemedyName, HomeRemedyDescription )
        Values ( @HomeRemedyName, @HomeRemedyDescription );
    SELECT @NewHomeRemedyID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[HomeRemedy]
[dbo].[usp.LogError]
[dbo].[uspPrintError]



[dbo].[usp_InsertHouseholdSourcesofLead]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@HouseholdItemName	varchar(50)	50	
@HouseholdItemDescription	varchar(512)	512	
@NewHouseholdItemID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new HouseholdSourcesofLead records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertHouseholdSourcesofLead]      -- usp_InsertHousehold-
SourcesofLead

    -- Add the parameters for the stored procedure here

    @HouseholdItemName varchar(50) = NULL,
    @HouseholdItemDescription varchar(512) = NULL,
    @NewHouseholdItemID int OUTPUT

    AS
    BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into HouseholdSourcesofLead ( HouseholdItemName, HouseholdItem-
Description )
        Values ( @HouseholdItemName, @HouseholdItemDescription );
    SELECT @NewHouseholdItemID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[HouseholdSourcesofLead]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertInsuranceProvider]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@InsuranceProviderName	varchar(50)	50	
@NewInsuranceProviderID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new InsuranceProvider records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertInsuranceProvider]      -- usp_InsertInsuranceProvider
    -- Add the parameters for the stored procedure here
    @InsuranceProviderName varchar(50) = NULL,
    @NewInsuranceProviderID int OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
```

```
DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into InsuranceProvider ( InsuranceProviderName ) --, HouseholdItem-
Description )

        Values ( @InsuranceProviderName ) -- , @HouseholdItemDescription
);

    SELECT @NewInsuranceProviderID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[InsuranceProvider]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertLab]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@LabName	varchar(50)	50	
@LabDescription	varchar(250)	250	
@New_Lab_Notes	varchar(3000)	3000	
@NewLabID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new Lab records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertLab]    -- usp_InsertLab
    -- Add the parameters for the stored procedure here

    @LabName varchar(50) = NULL,
    @LabDescription varchar(250) = NULL,
    @New_Lab_Notes varchar(3000) = NULL,
    @NewLabID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int, @NotesID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into Lab ( LabName, LabDescription )
        Values ( @LabName, @LabDescription );
    SELECT @NewLabID = SCOPE_IDENTITY();

    IF (@New_Lab_Notes IS NOT NULL)
        EXEC      [dbo].[usp_InsertLabNotes]
            @Lab_ID = @NewLabID,
            @Notes = @New_Lab_Notes,
            @InsertedNotesID = @NotesID OUTPUT
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Lab]
[dbo].[usp_InsertLabNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertLabNotes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@Lab_ID	int	4	
@Notes	varchar(3000)	3000	
@InsertedNotesID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150215
-- Description: stored procedure to insert Lab notes
-- =====

CREATE PROCEDURE [dbo].[usp_InsertLabNotes]
    -- Add the parameters for the stored procedure here

    @Lab_ID int = NULL,
    @Notes VARCHAR(3000) = NULL,
    @InsertedNotesID INT OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @ErrorLogID int

    -- Insert statements for procedure here

    BEGIN TRY -- update Lab information

```

```
INSERT INTO LabNotes (LabID, Notes)
    values (@Lab_ID, @Notes);
SET @InsertedNotesID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    -- inserting information in the ErrorLog.

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[LabNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertLab]

[dbo].[usp_InsertLanguage]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@LanguageName	varchar(50)	50	
@LANGUAGEID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20130506
-- Description: Stored Procedure to insert new Languages
-- =====

CREATE PROCEDURE [dbo].[usp_InsertLanguage]    -- usp_InsertLanguage "Italian"
    -- Add the parameters for the stored procedure here
    @LanguageName varchar(50),
    @LANGUAGEID int OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    -- Insert statements for procedure here

    DECLARE @DBNAME NVARCHAR(128), @ErrorLogID int;
    SET @DBNAME = DB_NAME();

```

Author: liam

```
BEGIN TRY
    if Exists (select LanguageName from language where LanguageName = @Language-
Name)
        BEGIN
            RAISERROR
                (N'The language: %s already exists.',
                11, -- Severity.

                1, -- State.

                @LanguageName);
        END

    INSERT into Language (LanguageName) Values (upper(@LanguageName))
    SET @LANGUAGEID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Language]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertMedium]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@MediumName	varchar(50)	50	
@MediumDescription	varchar(250)	250	
@TriggerLevel	int	4	
@TriggerLevelUnitsID	smallint	2	
@NewMediumID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new Medium records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertMedium] -- usp_InsertMedium
    -- Add the parameters for the stored procedure here
    @MediumName varchar(50) = NULL,
    @MediumDescription varchar(250) = NULL,
    @TriggerLevel int = NULL,
    @TriggerLevelUnitsID smallint = NULL,
    @NewMediumID int OUTPUT

AS
BEGIN
```

Author: liam

```

-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into Medium ( MediumName, MediumDescription, TriggerLevel, TriggerLevel-
UnitsID )
        Values ( @MediumName, @MediumDescription, @TriggerLevel, @Trigger-
LevelUnitsID );
    SELECT @NewMediumID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO

```

Uses

[dbo].[Medium]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertMediumSampleResults]	
---------------------------------------	--

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@MediumID	int	4	
@MediumSampleValue	numeric(9,4)	5	
@UnitsID	smallint	2	
@SampleLevelCategoryID	tinyint	1	
@MediumSampleDate	date	3	
@LabID	int	4	
@LabSubmissionDate	date	3	
@Notes	varchar(3000)	3000	
@IsAboveTriggerLevel	bit	1	
@NewMediumSampleResultsID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new MediumSampleResults records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertMediumSampleResults]    -- usp_InsertMedium-
Results

-- Add the parameters for the stored procedure here
```

```
@MediumID int = NULL,
@MediumSampleValue numeric(9,4) = NULL,
@UnitsID smallint = NULL,
@SampleLevelCategoryID tinyint = NULL,
@MediumSampleDate date = getdate,
@LabID int = NULL,
@LabSubmissionDate date = getdate,
@Notes varchar(3000) = NULL,
@IsAboveTriggerLevel bit = NULL,
@NewMediumSampleResultsID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int, @TriggerlevelUnitsID smallint, @TriggerLevel numeric(9,4),
    @NotesID int, @Notes_results int;
    -- Insert statements for procedure here

    -- See if Value is above Trigger Level - initially assume units are identical

    -- Determine Trigger level units and Trigger Level

    Select @TriggerLevel = M.TriggerLevel , @TriggerLevelUnitsID = M.TriggerLevelUnits-
    ID FROM MediumSampleresults AS MSR
        JOIN Medium as M on M.MediumID = MSR.MediumID
        JOIN Units AS TLU on M.TriggerLevelUnitsID = TLU.UnitsID;

    -- IF the units are the same,

    IF (@UnitsID = @TriggerlevelUnitsID )
    BEGIN
        print 'units are identical comparing values'
        IF ( @MediumSampleValue < @TriggerLevel )
            SET @IsAboveTriggerLevel = 0;
        ELSE
            SET @IsAboveTriggerLevel = 1;
    END
    ELSE
        print 'consider converting values to the same units'

    BEGIN TRY
        INSERT into MediumSampleResults ( MediumID, MediumSampleValue, UnitsID, Sample-
        LevelCategoryID, MediumSampleDate, LabID,
                                         LabSubmissionDate, IsAboveTriggerLevel )
            Values ( @MediumID, @MediumSampleValue, @UnitsID, @SampleLevel-
        CategoryID, @MediumSampleDate, @LabID,
                                         @LabSubmissionDate, @IsAboveTriggerLevel );
    END TRY
    BEGIN CATCH
        -- Handle errors
    END CATCH
END
```

```
SELECT @NewMediumSampleResultsID = SCOPE_IDENTITY();

IF (@Notes IS NOT NULL)
    EXEC    @Notes_results = [usp_InsertMediumSampleResultsNotes]
            @MediumSampleResults_ID = @NewMediumSampleResultsID,
            @Notes = @Notes,
            @InsertedNotesID = @NotesID OUTPUT

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END
GO
```

Uses

[dbo].[Medium]
[dbo].[MediumSampleResults]
[dbo].[Units]
[dbo].[usp_InsertMediumSampleResultsNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertMediumSampleResultsNotes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@MediumSampleResults_ID	int	4	
@Notes	varchar(3000)	3000	
@InsertedNotesID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150417
-- Description: stored procedure to insert MediumSampleResults notes
-- =====

CREATE PROCEDURE [dbo].[usp_InsertMediumSampleResultsNotes]
    -- Add the parameters for the stored procedure here
    @MediumSampleResults_ID int = NULL,
    @Notes VARCHAR(3000) = NULL,
    @InsertedNotesID INT OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @ErrorLogID int

    -- Insert statements for procedure here
    BEGIN TRY -- update MediumSample information
```

```
INSERT INTO MediumSampleResultsNotes (MediumSampleResultsID, Notes)
    values (@MediumSampleResults_ID, @Notes);
SET @InsertedNotesID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    -- inserting information in the ErrorLog.

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[MediumSampleResultsNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertMediumSampleResults]



[dbo].[usp_InsertNewBloodLeadTestResultsWebScreen]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@Person_ID	int	4	
@Sample_Date	date	3	
@Lab_Date	date	3	
@Blood_Lead_Result	numeric(4,1)	5	
@Flag	int	4	
@Test_Type	tinyint	1	
@Lab	varchar(50)	50	
@Lab_ID	int	4	
@Child_Status_Code	smallint	2	
@Child_Status_Date	date	3	
@Hemoglobin_Value	numeric(4,1)	5	
@DEBUG	bit	1	
@Blood_Test_Results_ID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20141217
-- Description: stored procedure to insert data retrieved from
--               the Blood Lead Test Results web screen
-- =====

CREATE PROCEDURE [dbo].[usp_InsertNewBloodLeadTestResultsWebScreen]
    -- Add the parameters for the stored procedure here
```

```
@Person_ID int = NULL,
@Sample_Date date = NULL,
@Lab_Date date = Null,
@Blood_Lead_Result numeric(4,1)= NULL, -- Is this Lead value?

@Flag INT = 365, -- flag follow up date

@Test_Type tinyint = NULL, -- SampleTypeID need to determine if/how new testTypes
are created

@Lab varchar(50) = NULL, -- is this necessary i think the lab should be selected
from a drop down with the option to add a new lab and an id should be passed?

@Lab_ID int = NULL,
@Child_Status_Code smallint = NULL, -- StatusID need to determine if/how new status-
Codes are created

@Child_Status_Date date = NULL,
@Hemoglobin_Value numeric(4,1) = NULL,
@DEBUG bit = 0,
@Blood_Test_Results_ID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @BloodTestResult_return_value int, @RetestDate_return_value int
        ,@Retest_Date date, @ChildStatusCode_return_value int, @ErrorLogID int;

    -- set default date if necessary

    IF (@Sample_Date is null)
    BEGIN
        set @Sample_Date = GetDate();
        RAISERROR ('Need to specify the SampleDate, setting to today by default', 5,
0);
    END

    IF (@Person_ID IS NULL)
    BEGIN
        RAISERROR ('Client name must be supplied', 11, -1);
        RETURN;
    END;
    BEGIN TRY
        EXEC    @BloodTestResult_return_value = [dbo].[usp_InsertBloodTestResults]
            @isBaseline = NULL,
            @PersonID = @Person_ID,
            @SampleDate = @Sample_Date,
            @LabSubmissionDate = @Lab_Date,
            @LeadValue = @Blood_Lead_Result,
            @LeadValueCategoryID = NULL,
```

```
        @HemoglobinValue = @Hemoglobin_Value,
        @HemoglobinValueCategoryID = NULL,
        @HematocritValueCategoryID = NULL,
        @LabID = @Lab_ID,
        @ClientStatusID = @Child_Status_Code,
        @BloodTestCosts = NULL,
        @sampleTypeID = @Test_Type,
        @New_Notes = NULL,
        @TakenAfterPropertyRemediationCompleted = NULL,
        @BloodTestResultID = @Blood_Test_Results_ID OUTPUT

--IF (@Child_Status_Code IS NOT NULL)

--BEGIN

--    IF (@Child_Status_Date IS NULL)

--        SELECT @Child_Status_Date = GetDate();

--    IF @DEBUG = 1

--        SELECT '@ChildStatusCode_return_value = [dbo].[usp_InsertPersono-
Status] @PersonID = @Person_ID, @StatusID = @Child_Status_Code, @StatusDate = @Sample_-
Date'

--                ,@Person_ID , @Child_Status_Code, @Sample_Date

--        EXEC      @ChildStatusCode_return_value = [dbo].[usp_InsertPersonoStatus]

--                @PersonID = @Person_ID,

--                @StatusID = @Child_Status_Code,

--                @StatusDate = @Sample_Date

--END

-- set the retest date based on integer value passed in as Flag

SET @Retest_Date = DATEADD(dd,@Flag,@Sample_Date);

-- update Person table with the new retest date

-- anyone with a blood test is a client

EXEC      @RetestDate_return_value = [dbo].[usp_upPerson]
        @Person_ID = @Person_ID
        , @New_RetestDate = @Retest_Date
        , @New_ClientStatusID = @Child_Status_Code
        , @New_isClient = 1;
```

```
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp_InsertBloodTestResults]
[dbo].[usp_upPerson]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_InsertNewClient-WebScreen

[dbo].[usp_InsertNewClientWebScreen]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@Family_ID	int	4	
@First_Name	varchar(50)	50	
@Middle_Name	varchar(50)	50	
@Last_Name	varchar(50)	50	
@Birth_Date	date	3	
@Gender_	char	1	
@Language_ID	tinyint	1	
@Ethnicity_ID	int	4	
@Moved_	bit	1	
@Travel	bit	1	
@Travel_Notes	varchar(3000)	3000	
@Out_of_Site	bit	1	
@Hobby_ID	smallint	2	
@Hobby_Notes	varchar(3000)	3000	
@Client_Notes	varchar(3000)	3000	
@Release_Notes	varchar(3000)	3000	
@is_Smoker	bit	1	
@Occupation_ID	smallint	2	
@Occupation_Start_Date	date	3	
@OverrideDuplicatePerson	bit	1	
@EmailAddress	varchar(320)	320	
@is_Client	bit	1	
@NursingMother	bit	1	
@NursingInfant	bit	1	
@Pregnant	bit	1	
@ClientID	int	4	Out

SQL Script

```
-- =====

-- Author:      Liam Thier

-- Create date: 20141115

-- Description: stored procedure to insert data from the Add a new client web page

-- =====

CREATE PROCEDURE [dbo].[usp_InsertNewClientWebScreen]
    -- Add the parameters for the stored procedure here

    @Family_ID int = NULL,
    @First_Name varchar(50) = NULL,
    @Middle_Name varchar(50) = NULL,
    @Last_Name varchar(50) = NULL,
    @Birth_Date date = NULL,
    @Gender_ char(1) = NULL,
    @Language_ID tinyint = NULL,
    @Ethnicity_ID int = NULL,
    @Moved_ bit = NULL,
    @Travel bit = NULL, --ForeignTravel REMOVE AFTE MOVING TO ADDNewFamilyWebScreen

    @Travel_Notes varchar(3000) = NULL, -- REMOVE AFTE MOVING TO ADDNewFamilyWebScreen

    @Out_of_Site bit = NULL,
    @Hobby_ID smallint = NULL,
    @Hobby_Notes varchar(3000) = NULL,
    @Client_Notes varchar(3000) = NULL,
    @Release_Notes varchar(3000) = NULL,
    @is_Smoker bit = NULL,
    @Occupation_ID smallint = NULL,
    @Occupation_Start_Date date = NULL,
    @OverrideDuplicatePerson bit = 0,
    @EmailAddress VARCHAR(320) = NULL,
    @is_Client bit = 1,
    @NursingMother bit = NULL,
    @NursingInfant bit = NULL,
    @Pregnant bit = NULL,
    @ClientID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    BEGIN
```

Author: liam

```
DECLARE @ErrorLogID int,
        @Ethnicity_return_value int,
        @PersontoFamily_return_value int,
        @PersontoLanguage_return_value int,
        @PersontoHobby_return_value int,
        @PersontoOccupation_return_value int,
        @PersontoEthnicity_return_value int;

-- If no family ID was passed in exit

IF (@Family_ID IS NULL)
BEGIN
    RAISERROR ('Family name must be supplied', 11, -1);
    RETURN;
END;

-- If the family doesn't exist, return an error

IF ((select FamilyID from family where FamilyID = @Family_ID) is NULL)
BEGIN
    DECLARE @ErrorString VARCHAR(3000);
    SET @ErrorString = 'Unable to associate non-existent family. Family does
not exist.';
    RAISERROR (@ErrorString, 11, -1);
    RETURN;
END

if (@Last_Name is null)
BEGIN
    select @Last_Name = Lastname from Family where FamilyID = @Family_ID
END

BEGIN TRY -- insert new person

    EXEC [dbo].[usp_InsertPerson]
        @FirstName = @First_Name,
        @MiddleName = @Middle_Name,
        @LastName = @Last_Name,
        @BirthDate = @Birth_Date,
        @Gender = @Gender_,
        @Moved = @Moved_,
        @ForeignTravel = @Travel,
        @EmailAddress = @EmailAddress,
        @OutofSite = @Out_of_Site,
        @New_Notes = @Client_Notes,
        @Hobby_Notes = @Hobby_Notes,
        @Release_Notes = @Release_Notes,
        @Travel_Notes = @Travel_Notes,
        @isSmoker = @is_Smoker,
        @isClient = @is_Client,
        @NursingMother = @NursingMother,
        @NursingInfant = @NursingInfant,
        @Pregnant = @Pregnant,
```

```
    @OverrideDuplicate = @OverrideDuplicatePerson,
    @PID = @ClientID OUTPUT;

    -- Associate person to Ethnicity

    IF (@Ethnicity_ID IS NOT NULL)
        EXEC    @Ethnicity_return_value = [dbo].[usp_InsertPersontoEthnicity]
                @PersonID = @ClientID,
                @EthnicityID = @Ethnicity_ID

    -- Associate person to family

    if (@Family_ID is not NULL)
        EXEC    @PersontoFamily_return_value = usp_InsertPersontoFamily
                @PersonID = @ClientID, @FamilyID = @Family_ID, @OUTPUT = @Personto-
Family_return_value OUTPUT;

    -- Associate person to language

    if (@Language_ID is not NULL)
        EXEC    @PersontoLanguage_return_value = usp_InsertPersontoLanguage
                @LanguageID = @Language_ID, @PersonID = @ClientID, @isPrimary-
Language = 1;

    -- associate person to Hobby

    if (@Hobby_ID is not NULL)
        EXEC    @PersontoHobby_return_value = usp_InsertPersontoHobby
                @HobbyID = @Hobby_ID, @PersonID = @ClientID;

    -- associate person to occupation

    if (@Occupation_ID is not NULL)
        EXEC    @PersontoOccupation_return_value = [dbo].[usp_InsertPersonto-
Occupation]
                @PersonID = @ClientID,
                @OccupationID = @Occupation_ID
    END TRY
    BEGIN CATCH -- insert person

        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before

        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
            BEGIN
                ROLLBACK TRANSACTION;
            END
    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
```

```
    RETURN ERROR_NUMBER()
END CATCH; -- insert new person

END

IF (@Family_ID is not NULL AND @PersontoFamily_return_value <> 0)
BEGIN
    RAISERROR ('Error associating person to family', 11, -1);
    RETURN;
END

IF (@Hobby_ID is not NULL AND @PersontoHobby_return_value <> 0)
BEGIN
    RAISERROR ('Error associating person to Hobby', 11, -1);
    RETURN;
END

IF (@Language_ID is not NULL AND @PersontoLanguage_return_value <> 0)
BEGIN
    RAISERROR ('Error associating person to language', 11, -1);
    RETURN;
END

IF (@Occupation_ID is not NULL and @PersontoOccupation_return_value <> 0)
BEGIN
    RAISERROR ('Error associating person to occupation', 11, -1);
    RETURN;
END
END
GO
```

Uses

[dbo].[Family]
[dbo].[usp_InsertPerson]
[dbo].[usp_InsertPersontoEthnicity]
[dbo].[usp_InsertPersontoFamily]
[dbo].[usp_InsertPersontoHobby]
[dbo].[usp_InsertPersontoLanguage]
[dbo].[usp_InsertPersontoOccupation]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertNewFamilyWebScreen]	
--------------------------------------	--

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@FamilyLastName	varchar(50)	50	
@Address_Line1	varchar(100)	100	
@Address_Line2	varchar(100)	100	
@CityName	varchar(50)	50	
@StateAbbr	char(2)	2	
@ZipCode	varchar(10)	10	
@Year_Built	date	3	
@Movein_Date	date	3	
@MoveOut_Date	date	3	
@Owner_id	int	4	
@is_Owner_Occupied	bit	1	
@is_Residential	bit	1	
@has_Peeling_Chipping_Paint	bit	1	
@is_Rental	bit	1	
@PrimaryPhone	bigint	8	
@PrimaryPhonePriority	tinyint	1	
@SecondaryPhone	bigint	8	
@SecondaryPhonePriority	tinyint	1	
@Language	tinyint	1	
@NumSmokers	tinyint	1	
@Pets	tinyint	1	
@Frequently_Wash_Pets	bit	1	
@Petsonandout	bit	1	
@FamilyNotes	varchar(3000)	3000	
@PropertyNotes	varchar(3000)	3000	
@Travel	bit	1	
@Travel_Notes	varchar(3000)	3000	
@Travel_Start_Date	varchar(3000)	3000	

@Travel_End_Date	varchar(3000)	3000	
@OverrideDuplicateProperty	bit	1	
@OverrideDuplicateFamilyPropertyAssociation	bit	1	
@OwnerContactInformation	varchar(1000)	1000	
@DEBUG	bit	1	
@FamilyID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20141115
-- Description: stored procedure to insert data from the Add a new family web page
-- =====
-- 20150102      Fixed bug with family/property association checking

CREATE PROCEDURE [dbo].[usp_InsertNewFamilyWebScreen]
    -- Add the parameters for the stored procedure here

    @FamilyLastName varchar(50) = NULL,
    @Address_Line1 varchar(100) = NULL,
    @Address_Line2 varchar(100) = NULL,
    @CityName varchar(50) = NULL,
    @StateAbbr char(2) = NULL,
    @ZipCode varchar(10) = NULL,
    @Year_Built date = NULL,
    @Movein_Date date = NULL,
    @MoveOut_Date date = NULL,
    @Owner_id int = NULL,
    @is_Owner_Occupied bit = NULL,
    @is_Residential bit = NULL,
    @has_Peeling_Chipping_Paint bit = NULL,
    @is_Rental bit = NULL,
    @PrimaryPhone bigint = NULL,
    @PrimaryPhonePriority tinyint = 1,
    @SecondaryPhone bigint = NULL,
    @SecondaryPhonePriority tinyint = 2,
    @Language tinyint = NULL,
    @NumSmokers tinyint = NULL,
    @Pets tinyint = NULL,
    @Frequently_Wash_Pets bit = NULL,
    @Petsinandout bit = NULL,
    @FamilyNotes varchar(3000) = NULL,
```

```
@PropertyNotes varchar(3000) = NULL,
@Travel bit = NULL,
@Travel_Notes varchar(3000) = NULL,
@Travel_Start_Date varchar(3000) = NULL,
@Travel_End_Date varchar(3000) = NULL,
@OverrideDuplicateProperty bit = 0,
@OverrideDuplicateFamilyPropertyAssociation bit = 0,
@OwnerContactInformation varchar(1000) = NULL,
@DEBUG BIT = 0,
@FamilyID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    IF (@FamilyLastName IS NULL
        AND @Address_Line1 IS NULL
        AND @Address_Line2 IS NULL
        AND @PrimaryPhone IS NULL
        AND @SecondaryPhone IS NULL)
    BEGIN
        RAISERROR ('You must supply at least one of the following: Family name, Street-
Number, Street Name, Street Suffix, Apartment number, Primary phone, or Secondary
phone', 11, -1);
        RETURN;
    END;
    BEGIN
        DECLARE @return_value int,
                @PhoneTypeID tinyint,
                @Family_return_value int,
                @PropID int, @LID tinyint,
                @InsertedFamilytoPropertyID int,
                @FamilytoProperty_return_value int,
                @Primaryphone_return_value int,
                @Secondaryphone_return_value int,
                @NewFamilyNotesID int,
                @TravelNotesReturnValue int,
                @ErrorLogID int;

        BEGIN TRY
            -- Insert the property address if it doesn't already exist

            -- Check if the property already exists, if it does, return the propertyID

            SELECT @PropID = [dbo].udf_DoesPropertyExist (
                @Address_Line1,
                @Address_Line2,
                @CityName,
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_InsertNewFamily-WebScreen

```
        @StateAbbr,
        @ZipCode
    )

--if (@is_Owner_Occupied = 1)

--    select @Owner_id = IDENT_CURRENT('Family')+1

select 'PropertyID' = @PropID;
if (@PropID is NULL)
BEGIN -- enter property

    EXEC [dbo].[usp_InsertProperty]
    @AddressLine1 = @Address_Line1,
    @AddressLine2 = @Address_Line2,
    @City = @CityName,
    @State = @StateAbbr,
    @Zipcode = @ZipCode,
    @New_PropertyNotes = @PropertyNotes,
    @YearBuilt = @Year_Built,
    @Ownerid = @Owner_id,
    @isOwnerOccupied = @is_Owner_Occupied,
    @isResidential = @is_Residential,
    @hasPeelingChippingPaint = @has_Peeling_Chipping_Paint,
    @isRental = @is_Rental,
    @OverrideDuplicate = @OverrideDuplicateProperty,
    @OwnerContactInformation = @OwnerContactInformation,
    @PropertyID = @PropID OUTPUT;
END -- enter property

-- Check if Family is already associated with property, if so, skip insert
and return warning:

if ((select count(PrimarypropertyID) from Family where LastName = @Family-
LastName and PrimaryPropertyID = @PropID) > 0)
BEGIN
    if (@OverrideDuplicateFamilyPropertyAssociation = 1)
    BEGIN
        -- update address in the future??
        RAISERROR ('Family is already associated with that Property', 11, -
1);
        RETURN;
    END
    ELSE
    BEGIN
        RAISERROR ('Family is already associated with that Property', 11, -
1);
        RETURN;
    END;
END
ELSE
BEGIN
```

Author: liam

```
EXEC      [dbo].[usp_InsertFamily]
@LastName = @FamilyLastName,
@NumberofSmokers = @NumSmokers,
@PrimaryLanguageID = @Language,
@Notes = @FamilyNotes,
@ForeignTravel = @Travel,
@New_Travel_Notes = @Travel_Notes,
@Travel_Start_Date = @Travel_Start_Date,
@Travel_End_Date = @Travel_End_Date,
@Pets = @Pets,
@Petssinandout = @Petssinandout,
@FrequentlyWashPets = @Frequently_Wash_Pets,
@PrimaryPropertyID = @PropID,
@FID = @FamilyID OUTPUT;
END

-- Associate family to property

EXEC @FamilytoProperty_return_value = [usp_InsertFamilytoProperty]
@FamilyID = @FamilyID,
@PropertyID = @PropID,
@StartDate = @Movein_Date,
@EndDate = @MoveOut_Date,
@DEBUG = @DEBUG,
@NewFamilytoPropertyID = @InsertedFamilytoPropertyID OUTPUT

if (@PrimaryPhone is not NULL)
BEGIN -- insert Primary Phone

    DECLARE @PrimaryPhoneNumberID_OUTPUT bigint;

    SELECT @PhoneTypeID = PhoneTypeID from PhoneNumberType where
PhoneNumberTypeName = 'Primary Phone';

    EXEC      @Primaryphone_return_value = [dbo].[usp_InsertPhoneNumber]
@PhoneNumber = @PrimaryPhone,
@PhoneNumberTypeID = @PhoneTypeID,
@DEBUG = @DEBUG,
@PhoneNumberID_OUTPUT = @PrimaryPhoneNumberID_OUTPUT OUTPUT

    EXEC      [dbo].[usp_InsertFamilytoPhoneNumber]
@FamilyID = @FamilyID,
@NumberPriority = @PrimaryPhonePriority,
@PhoneNumberID = @PrimaryPhoneNumberID_OUTPUT,
@DEBUG = @DEBUG
END -- insert Primary Phone

if (@SecondaryPhone is not NULL)
BEGIN -- insert Secondary Phone

    DECLARE @SecondaryPhoneNumberID_OUTPUT bigint;
```

```
        SELECT @PhoneTypeID = PhoneTypeID from PhoneNumberType where
PhoneNumberTypeName = 'Secondary Phone';

        EXEC    @Secondaryphone_return_value = [dbo].[usp_InsertPhoneNumber]
@PhoneNumber = @SecondaryPhone,
@PhoneNumberTypeID = @PhoneTypeID,
@DEBUG = @DEBUG,
@PhoneNumberID_OUTPUT = @SecondaryPhoneNumberID_OUTPUT OUTPUT

        EXEC    [dbo].[usp_InsertFamilytoPhoneNumber]
@FamilyID = @FamilyID,
@NumberPriority = @SecondaryPhonePriority,
@PhoneNumberID = @SecondaryPhoneNumberID_OUTPUT,
@DEBUG = @DEBUG

        END -- insert Secondary Phone

        END TRY
        BEGIN CATCH
            -- Call procedure to print error information.

            EXECUTE dbo.uspPrintError;

            -- Roll back any active or uncommittable transactions before
            -- inserting information in the ErrorLog.

            IF XACT_STATE() <> 0
            BEGIN
                ROLLBACK TRANSACTION;
            END

            EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
            RETURN ERROR_NUMBER()
        END CATCH;
    END
END

GO
```

Uses

[dbo].[Family]
[dbo].[PhoneNumberType]
[dbo].[usp_InsertFamily]
[dbo].[usp_InsertFamilytoPhoneNumber]
[dbo].[usp_InsertFamilytoProperty]
[dbo].[usp_InsertPhoneNumber]
[dbo].[usp_InsertProperty]
[dbo].[usp.LogError]

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_InsertNewFamily-
WebScreen

[dbo].[uspPrintError]
[dbo].[udf_DoesPropertyExist]

[dbo].[usp_InsertNewQuestionnaireWebScreen]	
---	--

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@Person_ID	int	4	
@QuestionnaireDate	date	3	
@PaintPeeling	bit	1	
@PaintDate	date	3	
@VisitRemodel	bit	1	
@RemodelDate	date	3	
@Vitamins	bit	1	
@HandWash	bit	1	
@Bottle	bit	1	
@NursingMother	bit	1	
@NursingInfant	bit	1	
@Pregnant	bit	1	
@Pacifier	bit	1	
@BitesNails	bit	1	
@EatsOutdoors	bit	1	
@NonFoodInMouth	bit	1	
@EatsNonFood	bit	1	
@SucksThumb	bit	1	
@Mouthing	bit	1	
@DaycareID	int	4	
@VisitsOldHomes	bit	1	
@DayCareNotes	varchar(3000)	3000	
@Source	int	4	
@QuestionnaireNotes	varchar(3000)	3000	
@Hobby1ID	smallint	2	
@Hobby2ID	smallint	2	
@Hobby3ID	smallint	2	
@HobbyNotes	varchar(3000)	3000	

@DEBUG	bit	1	
@Questionnaire_return_value	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20141208
-- Description: stored procedure to insert data
--               from the Lead Research Subject Questionnaire web page
-- =====

CREATE PROCEDURE [dbo].[usp_InsertNewQuestionnaireWebScreen]
    -- Add the parameters for the stored procedure here

    @Person_ID int = NULL,
    @QuestionnaireDate date = NULL,
    @PaintPeeling bit = NULL,
    @PaintDate date = NULL,
    @VisitRemodel bit = NULL,
    @RemodelDate date = NULL,
    @Vitamins bit = NULL,
    @HandWash bit = NULL,
    @Bottle bit = NULL,
    @NursingMother bit = NULL,
    @NursingInfant bit = NULL,
    @Pregnant bit = NULL,
    @Pacifier bit = NULL,
    @BitesNails bit = NULL,
    @EatsOutdoors bit = NULL,
    @NonFoodInMouth bit = NULL,
    @EatsNonFood bit = NULL,
    @SucksThumb bit = NULL,
    @Mouthing bit = NULL,
    @DaycareID int = NULL,
    @VisitsOldHomes bit = NULL,
    @DayCareNotes varchar(3000) = NULL,
    @Source int = NULL,
    @QuestionnaireNotes varchar(3000) = NULL,
    @Hobby1ID smallint = NULL,
    @Hobby2ID smallint = NULL,
    @Hobby3ID smallint = NULL,
    @HobbyNotes varchar(3000) = NULL,
    @DEBUG bit = 0,
    @Questionnaire_return_value int OUTPUT
```

```
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    IF (@PaintPeeling = '') SET @PaintPeeling = NULL;

    DECLARE @ErrorLogID int, @New_Notes varchar(3000), @PersontoHobby1_return_value int
        , @PersontoHobby2_return_value int, @PersontoHobby3_return_value int, @Hobby-
NotesID int;

    BEGIN TRY
        -- set default date if necessary

        IF (@QuestionnaireDate is null)
        BEGIN
            print 'Need to specify QuestionnaireDate, setting to today by default';
            set @QuestionnaireDate = GetDate();
        END

        IF (@Person_ID IS NULL)
        BEGIN
            RAISERROR ('Client name must be supplied', 11, -1);
            RETURN;
        END;

        -- Client ID must already exist in the database

        IF ( (select PersonID from person where personID = @Person_ID) is NULL)
        BEGIN
            RAISERROR ('Specified ClientID does not exist', 11, -1);
            RETURN;
        END

        SET @New_Notes = concat(@QuestionnaireNotes, ' ', @DayCareNotes);

        EXEC [dbo].[usp_InsertQuestionnaire]
            @PersonID = @Person_ID,
            @QuestionnaireDate = @QuestionnaireDate,
            @QuestionnaireDataSourceID = @Source,
            @VisitRemodeledProperty = @VisitRemodel,
            @PaintDate = @PaintDate,
            @RemodelPropertyDate = @RemodelDate,
            @isExposedtoPeelingPaint = @PaintPeeling,
            @isTakingVitamins = @Vitamins,
            @NursingMother = @NursingMother,
            @NursingInfant = @NursingInfant,
            @Pregnant = @Pregnant,
            @isUsingPacifier = @Pacifier,
            @isUsingBottle = @Bottle,
```

```
        @BitesNails = @BitesNails,
        @NonFoodEating = @EatsNonFood,
        @NonFoodinMouth = @NonFoodInMouth,
        @EatOutside = @EatsOutdoors,
        @Suckling = @SucksThumb,
        @Mouthing = @Mouthing,
        @FrequentHandWashing = @HandWash,
        @DaycareID = @DaycareID,
        @VisitsOldHomes = @VisitsOldHomes,
        @New_Notes = @New_Notes,
        @DEBUG = @DEBUG,
        @QuestionnaireID = @Questionnaire_return_value OUTPUT

    -- Set NursingMother, NursingInfant, and Pregnant attributes of the person
    -- according to the questionnaire

    -- anyone that completes a questionnaire is a client

    EXEC [dbo].[usp_upPerson] @Person_ID = @Person_ID, @New_NursingMother =
@NursingMother, @New_NursingInfant = @NursingInfant
                                , @New_Pregnant = @Pregnant, @New_isClient = 1, @DEBUG
= @DEBUG

    -- associate person to Hobby

    if (@Hobby1ID is not NULL)
        EXEC      @PersonontoHobby1_return_value = usp_InsertPersonontoHobby
                  @HobbyID = @Hobby1ID, @PersonID = @Person_ID;

    if (@Hobby2ID is not NULL)
        EXEC      @PersonontoHobby2_return_value = usp_InsertPersonontoHobby
                  @HobbyID = @Hobby2ID, @PersonID = @Person_ID;

    if (@Hobby3ID is not NULL)
        EXEC      @PersonontoHobby3_return_value = usp_InsertPersonontoHobby
                  @HobbyID = @Hobby3ID, @PersonID = @Person_ID;

    -- insert hobby notes

    IF (@HobbyNotes IS NOT NULL)
        EXEC      [dbo].[usp_InsertPersonHobbyNotes]
                  @Person_ID = @Person_ID,
                  @Notes = @HobbyNotes,
                  @InsertedNotesID = @HobbyNotesID OUTPUT
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.
```

```
IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Person]
[dbo].[usp_InsertPersonHobbyNotes]
[dbo].[usp_InsertPersontoHobby]
[dbo].[usp_InsertQuestionnaire]
[dbo].[usp_upPerson]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertOccupation]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@OccupationName	varchar(50)	50	
@OccupationDescription	varchar(256)	256	
@LeadExposure	bit	1	
@NewOccupationID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new Occupation records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertOccupation]    -- usp_InsertOccupation
    -- Add the parameters for the stored procedure here

    @OccupationName varchar(50) = NULL,
    @OccupationDescription varchar(256) = NULL,
    @LeadExposure bit = NULL,
    @NewOccupationID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into Occupation ( OccupationName, OccupationDescription, LeadExposure )
        Values ( @OccupationName, @OccupationDescription, @LeadExposure );
    SELECT @NewOccupationID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Occupation]
 [dbo].[usp.LogError]
 [dbo].[uspPrintError]

[dbo].[usp_InsertPerson]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@FirstName	varchar(50)	50	
@MiddleName	varchar(50)	50	
@LastName	varchar(50)	50	
@BirthDate	date	3	
@Gender	char	1	
@StatusID	smallint	2	
@ForeignTravel	bit	1	
@OutofSite	bit	1	
@EatsForeignFood	bit	1	
@EmailAddress	varchar(320)	320	
@RetestDate	datetime	8	
@Moved	bit	1	
@MovedDate	date	3	
@isClosed	bit	1	
@isResolved	bit	1	
@New_Notes	varchar(3000)	3000	
@Release_Notes	varchar(3000)	3000	
@Hobby_Notes	varchar(3000)	3000	
@Travel_Notes	varchar(3000)	3000	
@GuardianID	int	4	
@isSmoker	bit	1	
@isClient	bit	1	
@NursingMother	bit	1	
@NursingInfant	bit	1	
@Pregnant	bit	1	
@OverrideDuplicate	bit	1	
@PID	int	4	Out

SQL Script

```

-- =====
-- Author:      William Thier
-- Create date: 20130506
-- Description: Stored Procedure to insert new people records
-- =====

-- DROP PROCEDURE usp_InsertPerson

CREATE PROCEDURE [dbo].[usp_InsertPerson]    -- usp_InsertPerson
"Bonifacic", 'James', 'Marco', '19750205', 'M'

-- Add the parameters for the stored procedure here

@FirstName varchar(50) = NULL,
@MiddleName varchar(50) = NULL,
@LastName varchar(50) = NULL,
@BirthDate date = NULL,
@Gender char(1) = NULL,
@StatusID smallint = NULL,
@ForeignTravel bit = NULL,
@OutofSite bit = NULL,
@EatsForeignFood bit = NULL,
@EmailAddress VARCHAR(320) = NULL,
@RetestDate datetime = NULL,
@Moved bit = NULL,
@MovedDate date = NULL,
@isClosed bit = 0,
@isResolved bit = 0,
@New_Notes varchar(3000) = NULL,
@Release_Notes varchar(3000) = NULL,
@Hobby_Notes varchar(3000) = NULL,
@Travel_Notes varchar(3000) = NULL,
@GuardianID int = NULL,
@isSmoker bit = NULL,
@isClient bit = 1,
@NursingMother bit = 0,
@NursingInfant bit = 0,
@Pregnant bit = 0,
@OverrideDuplicate bit = 0,
@PID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

```

```

SET NOCOUNT ON;

DECLARE @ErrorLogID int, @NotesID int;

-- set default retest date if none specified

IF @RetestDate is null
    SET @RetestDate = DATEADD(yy,1,GetDate());

Select @PID = PersonID from Person where LastName = @LastName and FirstName =
@FirstName AND BirthDate = @BirthDate;
IF (@PID IS NOT NULL AND @OverrideDuplicate = 0)
BEGIN
    DECLARE @ErrorString VARCHAR(3000);
    SET @ErrorString ='Person appears to be a duplicate of personID: ' + cast(@PID
as varchar(256))
    RAISERROR (@ErrorString, 11, -1);
    RETURN;
END

-- Insert statements for procedure here

BEGIN TRY
    INSERT into person ( LastName, FirstName, MiddleName, BirthDate, Gender,
StatusID,
                        ForeignTravel, OutofSite, EatsForeignFood, Email-
Address, RetestDate,
                        Moved, MovedDate, isClosed, isResolved, GuardianID,
isSmoker,
                        isClient, NursingMother, NursingInfant, Pregnant)
    Values (@LastName, @FirstName, @MiddleName, @BirthDate, @Gender,
@StatusID,
                        @ForeignTravel, @OutofSite, @EatsForeignFood, @Email-
Address, @RetestDate,
                        @Moved, @MovedDate, @isClosed, @isResolved, @GuardianID,
@isSmoker,
                        @isClient, @NursingMother, @NursingInfant, @Pregnant);
    SET @PID = SCOPE_IDENTITY();

    IF (@New_Notes IS NOT NULL)
        EXEC      [dbo].[usp_InsertPersonNotes]
                    @Person_ID = @PID,
                    @Notes = @New_Notes,
                    @InsertedNotesID = @NotesID OUTPUT

    IF (@Release_Notes IS NOT NULL)
        EXEC      [dbo].[usp_InsertPersonReleaseNotes]
                    @Person_ID = @PID,
                    @Notes = @Release_Notes,
                    @InsertedNotesID = @NotesID OUTPUT

    IF (@Travel_Notes IS NOT NULL)
        EXEC      [dbo].[usp_InsertPersonTravelNotes]
                    @Person_ID = @PID,

```

```

@Notes = @Travel_Notes,
@InsertedNotesID = @NotesID OUTPUT

IF (@Hobby_Notes IS NOT NULL)
    EXEC      [dbo].[usp_InsertPersonHobbyNotes]
                @Person_ID = @PID,
                @Notes = @Hobby_Notes,
                @InsertedNotesID = @NotesID OUTPUT

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END
GO

```

Uses

[dbo].[Person]
[dbo].[usp_InsertPersonHobbyNotes]
[dbo].[usp_InsertPersonNotes]
[dbo].[usp_InsertPersonReleaseNotes]
[dbo].[usp_InsertPersonTravelNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertNewClientWebScreen]

[dbo].[usp_InsertPersonHobbyNotes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@Person_ID	int	4	
@Notes	varchar(3000)	3000	
@InsertedNotesID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150215
-- Description: stored procedure to insert PersonHobby notes
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersonHobbyNotes]
    -- Add the parameters for the stored procedure here

    @Person_ID int = NULL,
    @Notes VARCHAR(3000) = NULL,
    @InsertedNotesID INT OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @ErrorLogID int

    -- Insert statements for procedure here
```

Author: liam

```
BEGIN TRY -- update PersonHobby information

    INSERT INTO PersonHobbyNotes (PersonID, Notes)
        values (@Person_ID, @Notes);
    SET @InsertedNotesID = SCOPE_IDENTITY();

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    -- inserting information in the ErrorLog.

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[PersonHobbyNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertNewQuestionnaireWebScreen]
[dbo].[usp_InsertPerson]
[dbo].[usp_upPerson]

[dbo].[usp_InsertPersonNotes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@Person_ID	int	4	
@Notes	varchar(3000)	3000	
@InsertedNotesID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150215
-- Description: stored procedure to insert Person notes
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersonNotes]
    -- Add the parameters for the stored procedure here

    @Person_ID int = NULL,
    @Notes VARCHAR(3000) = NULL,
    @InsertedNotesID INT OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @ErrorLogID int

    -- Insert statements for procedure here

    BEGIN TRY -- update Person information

```

```
INSERT INTO PersonNotes (PersonID, Notes)
    values (@Person_ID, @Notes);
SET @InsertedNotesID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    -- inserting information in the ErrorLog.

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PersonNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertPerson]
[dbo].[usp_upPerson]

[dbo].[usp_InsertPersonReleaseNotes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@Person_ID	int	4	
@Notes	varchar(3000)	3000	
@InsertedNotesID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150215
-- Description: stored procedure to insert PersonRelease notes
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersonReleaseNotes]
    -- Add the parameters for the stored procedure here

    @Person_ID int = NULL,
    @Notes VARCHAR(3000) = NULL,
    @InsertedNotesID INT OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @ErrorLogID int

    -- Insert statements for procedure here
```

```
BEGIN TRY -- update PersonReleaseNotes information

    INSERT INTO PersonReleaseNotes (PersonID, Notes)
        values (@Person_ID, @Notes);
    SET @InsertedNotesID = SCOPE_IDENTITY();

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    -- inserting information in the ErrorLog.

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[PersonReleaseNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertPerson]
[dbo].[usp_upPerson]



[dbo].[usp_InsertPersonoAccessAgreement]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@AccessAgreementID	int	4
@AccessAgreementDate	date	3

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PersonoAccessAgreement records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersonoAccessAgreement]      -- usp_InsertPersono-
AccessAgreement

    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @AccessAgreementID int = NULL,
    @AccessAgreementDate date = NULL

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into PersonoAccessAgreement( PersonID, AccessAgreementID, Access-
    AgreementDate) --, EndDate)

        Values ( @PersonID, @AccessAgreementID, @AccessAgreementDate ) --
    , @EndDate);

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before

    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PersonoAccessAgreement]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertPersonstoDaycare]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@DaycareID	int	4
@StartDate	date	3
@EndDate	date	3
@DaycareNotes	varchar(3000)	3000

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PersonstoDaycare records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersonstoDaycare] -- usp_InsertPersonstoDaycare
    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @DaycareID int = NULL,
    @StartDate date = NULL,
    @EndDate date = NULL,
    @DaycareNotes varchar(3000) = NULL
AS
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_InsertPersonsto-Daycare

```
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
    -- Insert statements for procedure here

    BEGIN TRY
        INSERT into PersontoDaycare( PersonID, DaycareID, StartDate, EndDate)
            Values ( @PersonID, @DaycareID, @StartDate, @EndDate);
        --SELECT SCOPE_IDENTITY();

    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END
GO
```

Uses

[dbo].[PersontoDaycare]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertPersontoEmployer]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@EmployerID	int	4
@StartDate	date	3
@EndDate	date	3

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PersontoEmployer records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersontoEmployer]    -- usp_InsertPersontoEmployer
    -- Add the parameters for the stored procedure here
    @PersonID int = NULL,
    @EmployerID int = NULL,
    @StartDate date = NULL,
    @EndDate date = NULL
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into PersontoEmployer( PersonID, EmployerID, StartDate, EndDate)
        Values ( @PersonID, @EmployerID, @StartDate, @EndDate);
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PersontoEmployer]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertPersono-toEthnicity]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@EthnicityID	int	4

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PersonoEthnicity records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersonoEthnicity]    -- usp_InsertPersonoEthnicity
    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @EthnicityID int = NULL
    --@StartDate date = NULL,
    --@EndDate date = NULL

    AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    -- only provide support for association with a single ethnicity as per initial
    scope

    IF EXISTS (Select PersonID from PersontoEthnicity where PersonID = @PersonID)
        update PersontoEthnicity set EthnicityID = @EthnicityID where PersonID =
@PersonID;
    ELSE
        INSERT into PersontoEthnicity( PersonID, EthnicityID )
            Values ( @PersonID, @EthnicityID )

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PersontoEthnicity]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertNewClientWebScreen]
[dbo].[usp_upClientWebScreen]

[dbo].[usp_InsertPersontoFamily]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@PersonID	int	4	
@FamilyID	int	4	
@OUTPUT	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PersontoFamily records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersontoFamily] -- usp_InsertPersontoFamily
    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @FamilyID int = NULL,
    @OUTPUT int OUTPUT
    --@StartDate date = NULL,
    --@EndDate date = NULL

    AS
    BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int;

-- Insert statements for procedure here

BEGIN TRY
    INSERT into PersonstoFamily( PersonID, FamilyID ) --, StartDate, EndDate)
        Values ( @PersonID, @FamilyID ) -- , @StartDate, @EndDate);

    SELECT @OUTPUT = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PersonstoFamily]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertNewClientWebScreen]
[dbo].[usp_upClientWebScreen]

[dbo].[usp_InsertPersontoForeignFood]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@ForeignFoodID	int	4

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PersontoForeignFood records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersontoForeignFood]      -- usp_InsertPersontoForeignFood
-- Add the parameters for the stored procedure here

@PersonID int = NULL,
@ForeignFoodID int = NULL
--@StartDate date = NULL,
--@EndDate date = NULL

AS
BEGIN
-- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into PersontoForeignFood( PersonID, ForeignFoodID ) --, StartDate, End-
Date)

        Values ( @PersonID, @ForeignFoodID ) -- , @StartDate, @EndDate);

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before

    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PersontoForeignFood]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_InsertPersontoHobby

[dbo].[usp_InsertPersontoHobby]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@HobbyID	int	4

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PersontoHobby records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersontoHobby] -- usp_InsertPersontoHobby
    -- Add the parameters for the stored procedure here
    @PersonID int = NULL,
    @HobbyID int = NULL
    --@StartDate date = NULL,
    --@EndDate date = NULL

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

Author: liam

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into PersonstoHobby( PersonID, HobbyID ) --, StartDate, EndDate)
        Values ( @PersonID, @HobbyID ) -- , @StartDate, @EndDate);

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PersonstoHobby]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertNewClientWebScreen]
[dbo].[usp_InsertNewQuestionnaireWebScreen]
[dbo].[usp_upClientWebScreen]

[dbo].[usp_InsertPersonstoHomeRemedy]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@HomeRemedyID	int	4

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PersonstoHomeRemedy records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersonstoHomeRemedy] -- usp_InsertPersonstoHomeRemedy
    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @HomeRemedyID int = NULL
    --@StartDate date = NULL,
    --@EndDate date = NULL

    AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into PersonstoHomeRemedy( PersonID, HomeRemedyID ) --, StartDate, End-
Date)

        Values ( @PersonID, @HomeRemedyID ) -- , @StartDate, @EndDate);

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before

    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PersonstoHomeRemedy]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertPersontoInsurance]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@InsuranceID	smallint	2
@StartDate	date	3
@EndDate	date	3
@GroupID	varchar(20)	20

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PersontoInsurance records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersontoInsurance]    -- usp_InsertPersontoInsurance
    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @InsuranceID smallint = NULL,
    @StartDate date = NULL,
    @EndDate date = NULL,
    @GroupID varchar(20) = NULL

AS
```

```
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
    -- Insert statements for procedure here

    BEGIN TRY
        INSERT into PersonontoInsurance( PersonID, InsuranceID, StartDate, EndDate,
GroupID)
            Values ( @PersonID, @InsuranceID, @StartDate, @EndDate, @GroupID);
    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END
GO
```

Uses

[dbo].[PersonontoInsurance]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertPersontoLanguage]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@LanguageID	smallint	2
@isPrimaryLanguage	bit	1

SQL Script

```
-- =====

-- Author:      William Thier

-- Create date: 20140817

-- Description: Stored Procedure to insert new PersontoLanguage records

-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersontoLanguage]    -- usp_InsertPersontoLanguage

    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @LanguageID smallint = NULL,
    @isPrimaryLanguage bit = NULL
    --@StartDate date = NULL,
    --@EndDate date = NULL,
    --@GroupID varchar(20) = NULL
```

```
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
    -- Insert statements for procedure here

    BEGIN TRY
        IF EXISTS (SELECT PersonID from PersontoLanguage where PersonID = @PersonID and LanguageID = @LanguageID)
            BEGIN
                -- make sure there are no other primary languages

                IF (@isPrimaryLanguage = 1)
                    update PersontoLanguage set isPrimaryLanguage = 0 WHERE PersonID =
@PersonID AND LanguageID != @LanguageID AND isPrimaryLanguage = 1
                    update PersontoLanguage set isPrimaryLanguage = @isPrimaryLanguage WHERE
PersonID = @PersonID AND LanguageID = @LanguageID
                END
                ELSE
                    INSERT into PersontoLanguage( PersonID, LanguageID, isPrimaryLanguage ) --
StartDate, EndDate, GroupID)

                    Values ( @PersonID, @LanguageID, @isPrimaryLanguage ) -- @Start-
Date, @EndDate, @GroupID);

            END TRY
            BEGIN CATCH
                -- Call procedure to print error information.

                EXECUTE dbo.uspPrintError;

                -- Roll back any active or uncommittable transactions before
                -- inserting information in the ErrorLog.

                IF XACT_STATE() <> 0
                    BEGIN
                        ROLLBACK TRANSACTION;
                    END

                EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
                RETURN ERROR_NUMBER()
            END CATCH;
    END
GO
```

Uses

[dbo].[PersontoLanguage]
[dbo].[uspLogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertNewClientWebScreen]
[dbo].[usp_upClientWebScreen]

[dbo].[usp_InsertPersontoOccupation]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@OccupationID	smallint	2
@StartDate	date	3
@EndDate	date	3

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PersontoOccupation records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersontoOccupation]    -- usp_InsertPersontoOccupation
    -- Add the parameters for the stored procedure here
    @PersonID int = NULL,
    @OccupationID smallint = NULL,
    @StartDate date = NULL,
    @EndDate date = NULL
    --@GroupID varchar(20) = NULL

AS
BEGIN
```

```
-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @return_value int, @ErrorLogID int;

-- at the very least assume the start date is today

IF (@StartDate is NULL) SELECT @StartDate = GETDATE();

-- Insert statements for procedure here

BEGIN TRY
    INSERT into PersontoOccupation( PersonID, OccupationID, StartDate, EndDate)
        Values ( @PersonID, @OccupationID, @StartDate, @EndDate);
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER();
END CATCH;
END
GO
```

Uses

[dbo].[PersontoOccupation]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertNewClientWebScreen]
[dbo].[usp_upClientWebScreen]

[dbo].[usp_InsertPersontoPerson]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Person1ID	int	4
@Person2ID	smallint	2
@RelationshipType	int	4
@isGuardian	bit	1
@isPrimaryContact	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20150323
-- Description: Stored Procedure to insert
--               new PersontoPerson records how
--               they are related
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersontoPerson]    -- usp_InsertPersontoPerson
    -- Add the parameters for the stored procedure here
    @Person1ID int = NULL,
    @Person2ID smallint = NULL,
    @RelationshipType int = NULL,
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_InsertPersontoPerson

```
@isGuardian bit = NULL, -- True if P1 is guardian of P2

@isPrimaryContact bit = NULL
--@EndDate date = NULL,
--@GroupID varchar(20) = NULL

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
    -- Insert statements for procedure here

    BEGIN TRY
        INSERT into PersontoPerson( Person1ID, Person2ID, RelationshipTypeID, is-
Guardian, isPrimaryContact )
            Values ( @Person1ID, @Person2ID, @RelationShipType, @isGuardian,
@isPrimaryContact )

        -- Switch isGuardian information to update reciprocal relationship
        --IF (@isGuardian = 1) SET @isGuardian = 0;
        --ELSE SET @isGuardian = 1;

        --INSERT into PersontoPerson (Person1ID, Person2ID, isGuardian) values
(@Person2ID, @Person1ID, @isGuardian)

    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
```

Author: liam

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_InsertPersontoPerson

```
END  
GO
```

Uses

[dbo].[PersonstoPerson]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertPersono-toPhoneNumber]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@PhoneNumberID	int	4
@NumberPriority	tinyint	1

SQL Script

```
-- =====

-- Author:      William Thier

-- Create date: 20140817

-- Description: Stored Procedure to insert new Persono-toPhoneNumber records

-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersono-toPhoneNumber]    -- usp_InsertPersono-toPhoneNumber

    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @PhoneNumberID int = NULL,
    @NumberPriority tinyint = NULL

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from

    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into PersonoPhoneNumber( PersonID, PhoneNumberID, NumberPriority)
        Values ( @PersonID, @PhoneNumberID, @NumberPriority )
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PersonoPhoneNumber]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertPersontoProperty]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@PersonID	int	4	
@PropertyID	int	4	
@StartDate	date	3	
@EndDate	date	3	
@isPrimaryResidence	bit	1	
@NewPersontoPropertyID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PersontoProperty records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersontoProperty]    -- usp_InsertPersontoProperty

    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @PropertyID int = NULL,
    @StartDate date = NULL,
    @EndDate date = NULL,
    @isPrimaryResidence bit = NULL,
    @NewPersontoPropertyID int OUTPUT
```

```
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
    -- Insert statements for procedure here

    BEGIN TRY
        INSERT into PersonstoProperty( PersonID, PropertyID, StartDate, EndDate, is-
PrimaryResidence)
            Values ( @PersonID, @PropertyID, @StartDate, @EndDate, @isPrimary-
Residence )
        SELECT @NewPersonstoPropertyID = SCOPE_IDENTITY();
    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END
GO
```

Uses

[dbo].[PersonstoProperty]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertPersonoToTravelCountry]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@TravelCountryID	int	4
@StartDate	date	3
@EndDate	date	3

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PersonoToTravelCountry records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersonoToTravelCountry]    -- usp_InsertPersonoToTravel-
Country

    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @TravelCountryID int = NULL,
    @StartDate date = NULL,
    @EndDate date = NULL

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into PersonstoTravelCountry( PersonID, CountryID, StartDate, EndDate )
        Values ( @PersonID, @TravelCountryID, @StartDate, @EndDate )
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PersonToTravelCountry]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertPersonTravelNotes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@Person_ID	int	4	
@Notes	varchar(3000)	3000	
@InsertedNotesID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150215
-- Description: stored procedure to insert PersonTravel notes
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPersonTravelNotes]
    -- Add the parameters for the stored procedure here

    @Person_ID int = NULL,
    @Notes VARCHAR(3000) = NULL,
    @InsertedNotesID INT OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @ErrorLogID int

    -- Insert statements for procedure here
```

```
BEGIN TRY -- update PersonTravel information

    INSERT INTO PersonTravelNotes (PersonID, Notes)
        values (@Person_ID, @Notes);
    SET @InsertedNotesID = SCOPE_IDENTITY();

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    -- inserting information in the ErrorLog.

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[PersonTravelNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertPerson]
[dbo].[usp_upPerson]

[dbo].[usp_InsertPhoneNumber]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@CountryCode	tinyint	1	
@PhoneNumber	bigint	8	
@PhoneNumberTypeID	tinyint	1	
@DEBUG	bit	1	
@PhoneNumberID_OUTPUT	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PhoneNumber records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPhoneNumber]    -- usp_InsertPhoneNumber
    -- Add the parameters for the stored procedure here
    @CountryCode tinyint = 1,
    @PhoneNumber bigint = NULL,
    @PhoneNumberTypeID tinyint = NULL,
    @DEBUG bit = NULL,
    @PhoneNumberID_OUTPUT int OUTPUT
AS
```

```
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
    -- Insert statements for procedure here

    BEGIN TRY
        -- Determine if the phone number already exists

        SELECT @PhoneNumberID_OUTPUT = PhoneNumberID from PhoneNumber where PhoneNumber
= @PhoneNumber

        -- If the phone number doesn't exist, insert it and get the new id

        IF (@PhoneNumberID_OUTPUT IS NULL)
        BEGIN
            IF (@DEBUG = 1)
                SELECT 'INSERT into PhoneNumber ( CountryCode, PhoneNumber, PhoneNumber-
TypeID )'
                    Values ( @CountryCode, @PhoneNumber, @PhoneNumberTypeID ),',
@CountryCode, @PhoneNumber, @PhoneNumberTypeID

            INSERT into PhoneNumber ( CountryCode, PhoneNumber, PhoneNumberTypeID )
                Values ( @CountryCode, @PhoneNumber, @PhoneNumberTypeID );
            SELECT @PhoneNumberID_OUTPUT = SCOPE_IDENTITY();
        END
    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END
GO
```

Uses

[dbo].[PhoneNumber]
[dbo].[uspLogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertNewFamilyWebScreen]
[dbo].[usp_upFamilyWebScreen]

[dbo].[usp_InsertPhoneNumberType]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@PhoneNumberTypeName	varchar(50)	50	
@PhoneNumberTypeID_OUTPUT	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20141220
-- Description: Stored Procedure to insert new PhoneNumber records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPhoneNumberType] -- usp_InsertPhoneNumberType
    -- Add the parameters for the stored procedure here
    @PhoneNumberTypeName VarChar(50) = NULL,
    @PhoneNumberTypeID_OUTPUT int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
```

```
DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into PhoneNumberType ( PhoneNumberTypeName )
        Values ( @PhoneNumberTypeName );
    SELECT @PhoneNumberTypeID_OUTPUT = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PhoneNumberType]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertProperty]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@ConstructionTypeID	tinyint	1	
@AreaID	int	4	
@isInHistoricDistrict	bit	1	
@isRemodeled	bit	1	
@RemodelDate	date	3	
@isInCityLimits	bit	1	
@AddressLine1	varchar(100)	100	
@AddressLine2	varchar(100)	100	
@City	varchar(50)	50	
@State	char(2)	2	
@Zipcode	varchar(12)	12	
@YearBuilt	date	3	
@Ownerid	int	4	
@isOwnerOccupied	bit	1	
@ReplacedPipesFaucets	tinyint	1	
@TotalRemediationCosts	money	8	
@New_PropertyNotes	varchar(3000)	3000	
@isResidential	bit	1	
@isCurrentlyBeingRemodeled	bit	1	
@hasPeelingChippingPaint	bit	1	
@County	varchar(50)	50	
@isRental	bit	1	
@OverRideDuplicate	bit	1	
@OwnerContactInformation	varchar(1000)	1000	
@PropertyID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new property records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertProperty]    -- usp_InsertProperty
    -- Add the parameters for the stored procedure here

    @ConstructionTypeID tinyint = NULL,
    @AreaID int = NULL,
    @isinHistoricDistrict bit = NULL,
    @isRemodeled bit = NULL,
    @RemodelDate date = NULL,
    @isinCityLimits bit = NULL,
    @AddressLine1 varchar(100) = NULL,
    @AddressLine2 varchar(100) = NULL,
    @City varchar(50) = NULL,
    @State char(2) = NULL,
    @Zipcode varchar(12) = NULL,
    @YearBuilt date = NULL,
    @Ownerid int = NULL,
    @isOwnerOccupied bit = NULL,
    @ReplacedPipesFaucets tinyint = 0,
    @TotalRemediationCosts money = NULL,
    @New_PropertyNotes varchar(3000) = NULL,
    @isResidential bit = NULL,
    @isCurrentlyBeingRemodeled bit = NULL,
    @hasPeelingChippingPaint bit = NULL,
    @County varchar(50) = NULL,
    @isRental bit = NULL,
    @OverRideDuplicate bit = 1,
    @OwnerContactInformation varchar(1000) = NULL,
    @PropertyID int OUTPUT

    AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
```

```

DECLARE @ErrorLogID int, @NotesID int;
-- Insert statements for procedure here

BEGIN TRY
    -- Check if the property already exists, if it does, return the propertyID

    SELECT @PropertyID = [dbo].udf_DoesPropertyExist (
        @AddressLine1,
        @AddressLine2,
        @City,
        @State,
        @ZipCode
    )

    if (@PropertyID is NOT NULL)
        BEGIN
            if (@OverrideDuplicate = 0)
                BEGIN
                    DECLARE @ErrorMessage VARCHAR(3000);
                    SET @ErrorMessage = 'Property Address ' + @AddressLine1 + ', ' + @City +
', ' + @State + ', ' + @ZipCode
                        + ' ' + ' appears to be a duplicate of: ' +
cast(@PropertyID as varchar(30));
                    RAISERROR('@PropertyID exists: @AddressLine1, @City, @State, @ZipCode',
11, -1);
                RETURN;
            END
        END

    -- RETURN THE PropertyID of the matching property

    PRINT 'returning existing propertyID: ' + cast(@PropertyID as varchar);
    RETURN;
END

INSERT into property (ConstructionTypeID, AreaID, isinHistoricDistrict, is-
Remodeled, RemodelDate,
                     isinCityLimits, AddressLine1, AddressLine2, City,
[State], Zipcode,
                     YearBuilt, OwnerID, isOwnerOccupied, ReplacedPipes-
Faucets, TotalRemediationCosts,
                     isResidential, isCurrentlyBeingRemodeled, hasPeeling-
ChippingPaint, County, isRental
                     , OwnerContactInformation)
    Values ( @ConstructionTypeID, @AreaID, @isinHistoricDistrict, @is-
Remodeled, @RemodelDate,
                     @isinCityLimits, @AddressLine1, @AddressLine2, @City,
@State, @Zipcode,
                     @YearBuilt, @OwnerID, @isOwnerOccupied, @ReplacedPipes-
Faucets, @TotalRemediationCosts,
                     @isResidential, @isCurrentlyBeingRemodeled, @hasPeeling-
ChippingPaint, @County, @isRental
                     , @OwnerContactInformation);
    SET @PropertyID = SCOPE_IDENTITY();

    IF (@New_PropertyNotes IS NOT NULL)

```

```
EXEC      [dbo].[usp_InsertPropertyNotes]
          @Property_ID = @PropertyID,
          @Notes = @New_PropertyNotes,
          @InsertedNotesID = @NotesID OUTPUT
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Property]
[dbo].[usp_InsertPropertyNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]
[dbo].[udf_DoesPropertyExist]

Used By

[dbo].[usp_InsertNewFamilyWebScreen]
[dbo].[usp_upFamilyWebScreen]

[dbo].[usp_InsertPropertyNotes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@Property_ID	int	4	
@Notes	varchar(3000)	3000	
@InsertedNotesID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150215
-- Description: stored procedure to insert Property notes
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPropertyNotes]
    -- Add the parameters for the stored procedure here

    @Property_ID int = NULL,
    @Notes VARCHAR(3000) = NULL,
    @InsertedNotesID INT OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @ErrorLogID int

    -- Insert statements for procedure here
```

```
BEGIN TRY -- update Property information

    INSERT INTO PropertyNotes (PropertyID, Notes)
        values (@Property_ID, @Notes);
    SET @InsertedNotesID = SCOPE_IDENTITY();

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    -- inserting information in the ErrorLog.

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[PropertyNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertProperty]
[dbo].[usp_upProperty]

[dbo].[usp_InsertPropertySampleResults]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@isBaseline	bit	1	
@PropertyID	int	4	
@LabSubmissionDate	date	3	
@LabID	int	4	
@SampleTypeID	tinyint	1	
@Notes	varchar(3000)	3000	
@NewPropertySampleResultsID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PropertySampleResults records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPropertySampleResults]    -- usp_InsertPropertySample-
Results

-- Add the parameters for the stored procedure here

@isBaseline bit = NULL,
@propertyID int = NULL,
@LabSubmissionDate date = getdate,
@LabID int = NULL,
@SampleTypeID tinyint = NULL,
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_InsertProperty-SampleResults

```
@Notes varchar(3000) = NULL,
@NewPropertySampleResultsID int OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int, @ExistsPropertyID int, @NotesID int, @Notes_Results int;
    -- check if the property has a record in BloodTestResults Table

    select @ExistsPropertyID = PropertyID from PropertySampleResults

    -- Insert statements for procedure here

    BEGIN TRY
        -- Determine if this person already has an entry in BloodTestResults and set is-Baseline appropriately.

        IF (@isBaseline is NULL) -- nothing passed in for baseline

            BEGIN
                IF (@ExistsPropertyID is not NULL)
                    BEGIN
                        SET @isBaseline = 0;
                    END
                ELSE -- the person has no entry in BloodTestResults, this is a baseline
entry

                    BEGIN
                        SET @isBaseline = 1;
                    END
                END
            ELSE IF (@isBaseline = 0) -- this should not be a baseline entry according to
passed in argument

                BEGIN
                    IF (@ExistsPropertyID is NULL) -- the person does not have an entry in
BloodTestResults, this is a baseline entry

                        BEGIN
                            Set @isBaseline = 1;
                        END
                    END
                ELSE IF (@isBaseline = 1) -- this should be a baseline entry according to
passed in argument

                    BEGIN
                        IF (@ExistsPropertyID is not NULL) -- the person already has an entry in
BloodTestResults, this isn't a baseline entry
```

Author: liam

```
BEGIN
    Set @isBaseline = 0;
END
END

INSERT into PropertySampleResults ( isBaseline, PropertyID, LabSubmissionDate,
LabID,
                                         SampleTypeID )
VALUES ( @isBaseline, @PropertyID, @LabSubmissionDate, @LabID,
          @SampleTypeID );
SELECT @NewPropertySampleResultsID = SCOPE_IDENTITY();

IF (@Notes IS NOT NULL)
    EXEC    @Notes_results = [usp_InsertPropertySampleResultsNotes]
              @PropertySampleResults_ID = @NewPropertySampleResults-
ID,
              @Notes = @Notes,
              @InsertedNotesID = @NotesID OUTPUT
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END
GO
```

Uses

[dbo].[PropertySampleResults]
[dbo].[usp_InsertPropertySampleResultsNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertPropertySampleResultsNotes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@PropertySampleResults_ID	int	4	
@Notes	varchar(3000)	3000	
@InsertedNotesID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150417
-- Description: stored procedure to insert PropertySampleResults notes
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPropertySampleResultsNotes]
    -- Add the parameters for the stored procedure here

    @PropertySampleResults_ID int = NULL,
    @Notes VARCHAR(3000) = NULL,
    @InsertedNotesID INT OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @ErrorLogID int

    -- Insert statements for procedure here
```

```
BEGIN TRY -- update PropertySample information

    INSERT INTO PropertySampleResultsNotes (PropertySampleResultsID, Notes)
        values (@PropertySampleResults_ID, @Notes);
    SET @InsertedNotesID = SCOPE_IDENTITY();

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    -- inserting information in the ErrorLog.

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[PropertySampleResultsNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertPropertySampleResults]

[dbo].[usp_InsertPropertytoCleanupStatus]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PropertyID	int	4
@CleanupStatusID	tinyint	1
@CleanupStatusDate	date	3
@CostofCleanup	money	8

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PropertytoCleanupStatus records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPropertytoCleanupStatus] -- usp_InsertPropertyto-
CleanupStatus

    -- Add the parameters for the stored procedure here

    @PropertyID int = NULL,
    @CleanupStatusID tinyint = NULL,
    @CleanupStatusDate date = NULL,
    @CostofCleanup money = NULL

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into PropertytoCleanupStatus( PropertyID, CleanupStatusID, Cleanup-
StatusDate, CostofCleanup )
        Values ( @PropertyID, @CleanupStatusID, @CleanupStatusDate,
@CostofCleanup )
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PropertytoCleanupStatus]
[dbo].[usp.LogError]
[dbo].[uspPrintError]



[dbo].[usp_InsertPropertytoHouseholdSourcesofLead]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PropertyID	int	4
@HouseholdSourcesofLeadID	int	4

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PropertytoHouseholdSourcesofLead
records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPropertytoHouseholdSourcesofLead] -- usp_Insert-
PropertytoHouseholdSourcesofLead

    -- Add the parameters for the stored procedure here

    @PropertyID int = NULL,
    @HouseholdSourcesofLeadID int = NULL

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
```

```
DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into PropertytoHouseholdSourcesofLead( PropertyID, HouseholdSourcesofLeadID )
        Values ( @PropertyID, @HouseholdSourcesofLeadID )
    SELECT SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PropertytoHouseholdSourcesofLead]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertPropertytoMedium]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PropertyID	int	4
@MediumID	int	4
@MediumTested	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new PropertytoMedium records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertPropertytoMedium]    -- usp_InsertPropertytoMedium
    -- Add the parameters for the stored procedure here

    @PropertyID int = NULL,
    @MediumID int = NULL,
    @MediumTested bit = 1

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into PropertytoMedium( PropertyID, MediumID, MediumTested )
        Values ( @PropertyID, @MediumID, @MediumTested )
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PropertytoMedium]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertQuestionnaire]	
---------------------------------	--

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@PersonID	int	4	
@QuestionnaireDate	date	3	
@QuestionnaireDataSourceID	int	4	
@VisitRemodeledProperty	bit	1	
@PaintDate	date	3	
@RemodelPropertyDate	date	3	
@isExposedtoPeelingPaint	bit	1	
@isTakingVitamins	bit	1	
@NursingMother	bit	1	
@NursingInfant	bit	1	
@Pregnant	bit	1	
@isUsingPacifier	bit	1	
@isUsingBottle	bit	1	
@BitesNails	bit	1	
@NonFoodEating	bit	1	
@NonFoodinMouth	bit	1	
@EatOutside	bit	1	
@Suckling	bit	1	
@Mouthing	bit	1	
@FrequentHandWashing	bit	1	
@VisitsOldHomes	bit	1	
@DaycareID	int	4	
@New_Notes	varchar(3000)	3000	
@DEBUG	bit	1	
@QuestionnaireID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new Questionnaire records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertQuestionnaire]
    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @QuestionnaireDate date = getdate,
    @QuestionnaireDataSourceID int = NULL,
    @VisitRemodeledProperty bit = NULL,
    @PaintDate date = NULL,
    @RemodelPropertyDate date = NULL,
    @isExposedtoPeelingPaint bit = NULL,
    @isTakingVitamins bit = NULL,
    @NursingMother bit = NULL,
    @NursingInfant bit = NULL,
    @Pregnant bit = NULL,
    @isUsingPacifier bit = NULL,
    @isUsingBottle bit = NULL,
    @BitesNails bit = NULL,
    @NonFoodEating bit = NULL,
    @NonFoodinMouth bit = NULL,
    @EatOutside bit = NULL,
    @Suckling bit = NULL,
    @Mouthing bit = NULL,
    @FrequentHandWashing bit = NULL,
    @VisitsOldHomes bit = NULL,
    @DaycareID int = NULL,
    @New_Notes varchar(3000) = NULL,
    @DEBUG bit = NULL,
    @QuestionnaireID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
```

```
DECLARE @ErrorLogID int, @NotesID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into Questionnaire ( PersonID, QuestionnaireDate, QuestionnaireDataSourceID, VisitRemodeledProperty, PaintDate, RemodelPropertyDate,
                                isExposedtoPeelingPaint, isTakingVitamins, NursingMother, NursingInfant, Pregnant, isUsingPacifier, isUsingBottle,
                                Bitesnails, NonFood Eating, NonFoodinMouth, EatOutside, Suckling, Mouthing, FrequentHandWashing,
                                VisitsOldHomes, DaycareID )
    Values ( @PersonID, @QuestionnaireDate, @QuestionnaireDataSourceID, @VisitRemodeledProperty, @PaintDate, @RemodelPropertyDate,
              @isExposedtoPeelingPaint, @isTakingVitamins, @NursingMother, @NursingInfant, @Pregnant, @isUsingPacifier, @isUsingBottle,
              @Bitesnails, @NonFood Eating, @NonFoodinMouth, @EatOutside, @Suckling, @Mouthing, @FrequentHandWashing,
              @VisitsOldHomes, @DaycareID );
    SELECT @QuestionnaireID = SCOPE_IDENTITY();

    IF (@New_Notes IS NOT NULL)
        EXEC      [dbo].[usp_InsertQuestionnaireNotes]
                    @Questionnaire_ID = @QuestionnaireID,
                    @Notes = @New_Notes,
                    @InsertedNotesID = @NotesID OUTPUT
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Questionnaire]
[dbo].[usp_InsertQuestionnaireNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_Insert-Questionnaire

Used By

[dbo].[usp_InsertNewQuestionnaireWebScreen]



[dbo].[usp_InsertQuestionnaireNotes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@Questionnaire_ID	int	4	
@Notes	varchar(3000)	3000	
@InsertedNotesID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150215
-- Description: stored procedure to insert Questionnaire notes
-- =====

CREATE PROCEDURE [dbo].[usp_InsertQuestionnaireNotes]
    -- Add the parameters for the stored procedure here
    @Questionnaire_ID int = NULL,
    @Notes VARCHAR(3000) = NULL,
    @InsertedNotesID INT OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @ErrorLogID int

    -- Insert statements for procedure here
    BEGIN TRY -- update Questionnaire information
```

```
INSERT INTO QuestionnaireNotes (QuestionnaireID, Notes)
    values (@Questionnaire_ID, @Notes);
SET @InsertedNotesID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    -- inserting information in the ErrorLog.

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[QuestionnaireNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertQuestionnaire]
[dbo].[usp_upQuestionnaire]

[dbo].[usp_InsertRemediation]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@RemediationApprovalDate	date	3	
@RemediationStartDate	date	3	
@RemediationEndDate	date	3	
@PropertyID	int	4	
@RemediationActionPlanID	int	4	
@AccessAgreementID	int	4	
@FinalRemediationReportFile	varbinary(max)	max	
@FinalRemediationReportDate	date	3	
@RemediationCost	money	8	
@OneYearRemediationCompleteDate	date	3	
@Notes	varchar(3000)	3000	
@OneYearRemediationComplete	bit	1	
@NewRemediationID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new Remediation records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertRemediation]    -- usp_InsertRemediation
```

```

-- Add the parameters for the stored procedure here

@RemediationApprovalDate date = getdate,
@RemediationStartDate date = NULL,
@RemediationEndDate date = NULL,
@PropertyID int = NULL,
@RemediationActionPlanID int = NULL,
@AccessAgreementID int = NULL,
@FinalRemediationReportFile varbinary(max) = NULL,
@FinalRemediationReportDate date = Null,
@RemediationCost money = NULL,
@OneYearRemediationCompleteDate date = NULL,
@Notes varchar(3000) = NULL,
@OneYearRemediationComplete bit = NULL,
@NewRemediationID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int, @NotesID int, @RemediationNotes_return int;
    -- Insert statements for procedure here

    BEGIN TRY
        INSERT into Remediation ( RemediationApprovalDate, RemediationStartDate,
        RemediationEndDate, PropertyID
                               , RemediationActionPlanID, AccessAgreementID, Final-
        RemediationReportFile, FinalRemediationReportDate
                               , RemediationCost, OneYearRemediationCompleteDate,
        OneYearRemediationComplete )
            Values ( @RemediationApprovalDate, @RemediationStartDate,
        @RemediationEndDate, @PropertyID
                               , @RemediationActionPlanID, @AccessAgreementID, @Final-
        RemediationReportFile, @FinalRemediationReportDate
                               , @RemediationCost, @OneYearRemediationCompleteDate, @One-
        YearRemediationComplete );
        SELECT @NewRemediationID = SCOPE_IDENTITY();

        IF (@Notes IS NOT NULL)
            EXEC      [dbo].[usp_InsertRemediationNotes]
                        @Remediation_ID = @NewRemediationID,
                        @Notes = @Notes,
                        @InsertedNotesID = @NotesID OUTPUT
    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before

```

```
-- inserting information in the ErrorLog.

IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Remediation]
[dbo].[usp_InsertRemediationNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertRemediationActionPlan]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@RemediationActionPlanApprovalDate	date	3	
@HomeOwnerConsultationDate	date	3	
@ContractorCompletedInvestigationDate	date	3	
@EnvironmentalInvestigationID	int	4	
@RemediationActionPlanFinalReportSubmissionDate	date	3	
@RemediationActionPlanFile	varbinary(max)	max	
@PropertyID	int	4	
@NewRemediationActionPlanID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new RemediationActionPlan records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertRemediationActionPlan]    -- usp_InsertRemediation-
ActionPlan

    -- Add the parameters for the stored procedure here

    @RemediationActionPlanApprovalDate date = getdate,
    @HomeOwnerConsultationDate date = NULL,
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_InsertRemediationActionPlan

```
@ContractorCompletedInvestigationDate date = NULL,
@EnvironmentalInvestigationID int = NULL,
@RemediationActionPlanFinalReportSubmissionDate date = NULL,
@RemediationActionPlanFile varbinary(max) = NULL,
@PropertyID int = NULL,
@NewRemediationActionPlanID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int;
    -- Insert statements for procedure here

    BEGIN TRY
        INSERT into RemediationActionPlan ( RemediationActionPlanApprovalDate, Home-
OwnerConsultationDate, ContractorCompletedInvestigationDate
                                         , EnvironmentalInvestigationID,
RemediationActionPlanFinalReportSubmissionDate,
                                         RemediationActionPlanFile, PropertyID )
            Values ( @RemediationActionPlanApprovalDate, @HomeOwner-
ConsultationDate, @ContractorCompletedInvestigationDate
                                         , @EnvironmentalInvestigationID, @RemediationActionPlan-
FinalReportSubmissionDate
                                         , @RemediationActionPlanFile, @PropertyID );
        SELECT @NewRemediationActionPlanID = SCOPE_IDENTITY();
    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END
GO
```

Author: liam

Uses

[dbo].[RemediationActionPlan]

[dbo].[uspLogError]

[dbo].[uspPrintError]

[dbo].[usp_InsertRemediationNotes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@Remediation_ID	int	4	
@Notes	varchar(3000)	3000	
@InsertedNotesID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150417
-- Description: stored procedure to insert Remediation notes
-- =====

CREATE PROCEDURE [dbo].[usp_InsertRemediationNotes]
    -- Add the parameters for the stored procedure here

    @Remediation_ID int = NULL,
    @Notes VARCHAR(3000) = NULL,
    @InsertedNotesID INT OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @ErrorLogID int

    -- Insert statements for procedure here
```

Author: liam

```
BEGIN TRY -- update Remediation information

    INSERT INTO RemediationNotes (RemediationID, Notes)
        values (@Remediation_ID, @Notes);
    SET @InsertedNotesID = SCOPE_IDENTITY();

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    -- inserting information in the ErrorLog.

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[RemediationNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertRemediation]

[dbo].[usp_InsertSampleLevelCategory]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@SampleLevelCategoryName	varchar(20)	20	
@SampleLevelCategoryDescription	varchar(256)	256	
@NewSampleLevelCategoryID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new SampleLevelCategory records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertSampleLevelCategory]    -- usp_InsertSampleLevelCategory
-- Add the parameters for the stored procedure here

@SampleLevelCategoryName varchar(20) = NULL,
@SampleLevelCategoryDescription varchar(256) = NULL,
@NewSampleLevelCategoryID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into SampleLevelCategory ( SampleLevelCategoryName, SampleLevelCategory-
Description )
        Values ( @SampleLevelCategoryName, @SampleLevelCategoryDescription
);
    SELECT @NewSampleLevelCategoryID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[SampleLevelCategory]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertSampleType]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@SampleTypeName	varchar(20)	20	
@SampleTypeDescription	varchar(256)	256	
@NewSampleTypeID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new SampleType records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertSampleType]    -- usp_InsertSampleType
    -- Add the parameters for the stored procedure here
    @SampleTypeName varchar(20) = NULL,
    @SampleTypeDescription varchar(256) = NULL,
    @NewSampleTypeID int OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into SampleType ( SampleTypeName, SampleTypeDescription )
        Values ( @SampleTypeName, @SampleTypeDescription );
    SELECT @NewSampleTypeID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[SampleType]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertStatus]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@StatusName	varchar(20)	20	
@StatusDescription	varchar(256)	256	
@NewStatusID	int	4	Out

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to insert new Status records
-- =====

CREATE PROCEDURE [dbo].[usp_InsertStatus] -- usp_InsertStatus
    -- Add the parameters for the stored procedure here
    @StatusName varchar(20) = NULL,
    @StatusDescription varchar(256) = NULL,
    @NewStatusID int OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @ErrorLogID int;
-- Insert statements for procedure here

BEGIN TRY
    INSERT into Status ( StatusName, StatusDescription )
        Values ( @StatusName, @StatusDescription );
    SELECT @NewStatusID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_InsertTravelNotes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@Family_ID	int	4	
@Travel_Notes	varchar(3000)	3000	
@Start_Date	date	3	
@End_Date	date	3	
@InsertedNotesID	int	4	Out

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150319
-- Description: stored procedure to insert Travel notes
-- =====

CREATE PROCEDURE [dbo].[usp_InsertTravelNotes]
    -- Add the parameters for the stored procedure here

    @Family_ID int = NULL,
    @Travel_Notes VARCHAR(3000) = NULL,
    @Start_Date date = NULL,
    @End_Date date = NULL,
    @InsertedNotesID INT OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;
DECLARE @ErrorLogID int

-- Insert statements for procedure here

BEGIN TRY -- update Property information

    INSERT INTO TravelNotes (FamilyID, Notes, StartDate, EndDate)
        values (@Family_ID, @Travel_Notes, @Start_Date, @End_Date);
    SET @InsertedNotesID = SCOPE_IDENTITY();
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    -- inserting information in the ErrorLog.

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[TravelNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertFamily]

[dbo].[usp_SLAllBloodTestResults]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Person_ID	int	4
@Min_Lead_Value	numeric(4,1)	5
@Max_Lead_Value	numeric(4,1)	5
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20141222
-- Description: select blood test results
--               optionally only return for a specific
--               client
-- =====

CREATE PROCEDURE [dbo].[usp_SLAllBloodTestResults]
    -- Add the parameters for the stored procedure here

    @Person_ID int = NULL,
    @Min_Lead_Value numeric(4,1) = NULL,
    @Max_Lead_Value numeric(4,1) = NULL,
    @DEBUG bit = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @spexecutesqlStr      NVARCHAR(4000), @OrderBy NVARCHAR(500),
        @Recompile BIT = 1, @ErrorLogID int;
BEGIN
    -- Insert statements for procedure here

    SELECT @spexecutesqlStr = N'SELECT ''ClientID'' = [P].[personid], ''LastName'' =
[P].[LastName], [P].[FirstName], ''BirthDate'' = [P].[BirthDate]
        , [BTR].[SampleDate], ''Pb_ug_Per_dl'' = [BTR].[Lead-
Value]
        , ''Hb_g_Per_dl'' = [BTR].[HemoglobinValue], ''Retest-
BL'' = [P].[RetestDate]
        , ''Closed'' = [P].[isClosed] , ''Moved'' =
[P].[Moved], ''Movedate'' = [P].[MovedDate]
        from [Person] [P]
        join [BloodTestResults] [BTR] on [P].[PersonID] =
[BTR].[PersonID]
        WHERE [P].[isClient] = 1'

    IF @Person_ID IS NOT NULL
    BEGIN
        SELECT @spexecutesqlStr = @spexecutesqlStr + N' AND [p].[PersonID] = @PersonID'
        SET @OrderBy = ' ORDER BY [BTR].[LeadValue], [BTR].[SampleDate] desc'
    END

    IF @Min_Lead_Value IS NOT NULL
    BEGIN
        SELECT @spexecutesqlStr = @spexecutesqlStr + N' AND [BTR].[LeadValue] >= @Min-
LeadValue'
    END

    IF @Max_Lead_Value IS NOT NULL
    BEGIN
        SELECT @spexecutesqlStr = @spexecutesqlStr + N' AND [BTR].[LeadValue] < @Max-
LeadValue'
    END

    IF @Person_ID is NULL
    BEGIN
        SELECT @spexecutesqlStr = @spexecutesqlStr;
        SET @OrderBy = N' ORDER BY [p].[LastName], [P].[PersonID] ASC, [BTR].[Sample-
Date] DESC';
    END

    SELECT @spexecutesqlStr = @spexecutesqlStr + @OrderBy

    IF ( (@Person_ID IS NULL) AND (@Min_Lead_Value IS NULL) )
        SET @Recompile = 0;

    IF @Recompile = 1
```

```
SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    -- If debugging print out query

    IF (@DEBUG = 1)
        SELECT @spexecutesqlStr, 'PersonID' = @Person_ID, 'MinLeadValue' = @Min_-
Lead_Value, 'MaxLeadValue' = @Max_Lead_Value, 'Recompile' = @Recompile;

    EXEC [sp_executesql] @spexecutesqlStr
    , N'@PersonID int,@MinLeadValue numeric(4,1), @MaxLeadValue numeric(4,1)'
    , @PersonID = @Person_ID, @MinLeadValue = @Min_Lead_Value, @MaxLeadValue =
@Max_Lead_Value;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SLAllBloodTestResults2]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Person_ID	int	4
@Min_Lead_Value	numeric(4,1)	5
@Max_Lead_Value	numeric(4,1)	5
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20141222
-- Description: select blood test results
--               optionally only return for a specific
--               client
-- =====

CREATE PROCEDURE [dbo].[usp_SLAllBloodTestResults2]
    -- Add the parameters for the stored procedure here

    @Person_ID int = NULL,
    @Min_Lead_Value numeric(4,1) = NULL,
    @Max_Lead_Value numeric(4,1) = NULL,
    @DEBUG bit = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @spexecutesqlStr      NVARCHAR(4000), @OrderBy NVARCHAR(500),
        @Recompile BIT = 1, @ErrorLogID int;

-- Insert statements for procedure here

SET FMTONLY OFF
SELECT @spexecutesqlStr = N'SELECT ''ClientID'' = [P].[personid], ''LastName'' =
[P].[LastName], ''BirthDate'' = [P].[BirthDate]
, ''Hb_g_per_dl'' = [BTR].[HemoglobinValue], ''RetestDate'' = [P].[RetestDate]
, ''Close'' = [P].[isClosed], ''Moved'' = [P].[Moved],
''Movedate'' = [P].[MovedDate]
from [Person] [P]
join [BloodTestResults] [BTR] on [P].[PersonID] =
[BTR].[PersonID]
WHERE 1 = 1'

IF @Person_ID IS NOT NULL
BEGIN
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' AND [p].[PersonID] = @PersonID'
    SET @OrderBy = ' ORDER BY [BTR].[LeadValue], [BTR].[SampleDate] desc'
END

IF @Min_Lead_Value IS NOT NULL
BEGIN
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' AND [BTR].[LeadValue] >= @Min-
LeadValue'
END

IF @Max_Lead_Value IS NOT NULL
BEGIN
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' AND [BTR].[LeadValue] < @Max-
LeadValue'
END

IF @Person_ID is NULL
BEGIN
    SELECT @spexecutesqlStr = @spexecutesqlStr,
           SET @OrderBy = N' ORDER BY [BTR].[Leadvalue], [p].[LastName], [P].[PersonID]
ASC, [BTR].[SampleDate] DESC';
END

SELECT @spexecutesqlStr = @spexecutesqlStr + @OrderBy

IF ( (@Person_ID IS NULL) AND (@Min_Lead_Value IS NULL) )
    SET @Recompile = 0;

IF @Recompile = 1
```

```
SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION (RECOMPILE)';

BEGIN TRY
    -- If debugging print out query

    IF (@DEBUG = 1)
        SELECT @spexecutesqlStr, 'PersonID' = @Person_ID, 'MinLeadValue' = @Min_-
Lead_Value, 'MaxLeadValue' = @Max_Lead_Value, 'Recompile' = @Recompile;

    EXEC [sp_executesql] @spexecutesqlStr
        , N'@PersonID int,@MinLeadValue numeric(4,1), @MaxLeadValue numeric(4,1)'
        , @PersonID = @Person_ID, @MinLeadValue = @Min_Lead_Value, @MaxLeadValue =
@Max_Lead_Value;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SLAllBloodTestResultsMetaData]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Person_ID	int	4
@Min_Lead_Value	numeric(9,4)	5
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20141222
-- Description: select blood test results
--               optionally only return for a specific
--               client
-- =====

CREATE PROCEDURE [dbo].[usp_SLAllBloodTestResultsMetaData]
    -- Add the parameters for the stored procedure here

    @Person_ID int = NULL,
    @Min_Lead_Value numeric(9,4) = NULL,
    @DEBUG bit = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @spexecutesqlStr      NVARCHAR(4000), @OrderBy NVARCHAR(500),
        @Recompile BIT = 1, @ErrorLogID int;
BEGIN
    -- Insert statements for procedure here

    SELECT 'ClientID' = [P].[personid], 'LastName' = [P].[LastName], 'BirthDate' =
[P].[BirthDate]
        , [BTR].[SampleDate], 'Pb_ug_Per_dl' = [BTR].[LeadValue]
        , 'Hb_g_Per_dl' = [BTR].[HemoglobinValue], 'RetestBL' =
DATEADD(yy,1,sampledate)
        , 'RetestHB' = DATEADD(yy,1,sampledate), 'Close' = [P].[isClosed],
'Moved' = [P].[Moved]
        , 'Movedate' = [P].[MovedDate]
    from [Person] [P]
    join [BloodTestResults] [BTR] on [P].[PersonID] = [BTR].[PersonID]
    WHERE 1 = 0
END
END

GO
```

Uses

[dbo].[BloodTestResults]
[dbo].[Person]

[dbo].[usp_SIChildStatus]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@TargetType	varchar(50)	50
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150215
-- Description:    returns valid status codes for passed in type - Child
-- =====

CREATE PROCEDURE [dbo].[usp_SIChildStatus]
    -- Add the parameters for the stored procedure here

    @TargetType varchar(50) = NULL,
    @DEBUG     BIT    = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @spexecutesqlStr NVARCHAR(3000)

    select @spexecutesqlStr ='

        -- Insert statements for procedure here
    
```

```
SELECT [TS].[StatusName], [TS].[StatusID] from [TargetStatus] AS [TS]
where 1 = 1 AND TargetType = 'Person'

END
GO
```

Uses

[dbo].[TargetStatus]

[dbo].[usp_SIClientFollowUp]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150715
-- Description: stored procedure to list family members
-- =====

CREATE PROCEDURE [dbo].[usp_SIClientFollowUp]
    -- Add the parameters for the stored procedure here

    @StartDate date = NULL,
    @EndDate date = NULL
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @spexecutesQLStr NVARCHAR(4000)
        , @Recompile BIT = 1, @ErrorLogID int;

    --IF (@FamilyID IS NULL)

    --BEGIN
```

```

--      RAISERROR ('You must supply at least one parameter.', 11, -1);

--      RETURN;

--END;

SELECT @spexecutesQLStr =
N'SELECT [P].[PersonID],[P].[LastName],[P].[Firstname],[P].[RetestDate]   from
[person] as [p]
      where 1=1';

IF (@StartDate IS NOT NULL)
SELECT @spexecutesQLStr = @spexecutesQLStr
+ N' AND [P].[RetestDate] >= @StartDate';

IF (@EndDate IS NOT NULL)
SELECT @spexecutesQLStr = @spexecutesQLStr
+ N' AND [P].[RetestDate] < @EndDate';
SELECT @spexecutesQLStr = @spexecutesQLStr + ' order by [P].[RetestDate] ASC'

IF (DateDiff(yy,@EndDate,@StartDate) > 4)
SET @Recompile = 0;

IF @Recompile = 1
SELECT @spexecutesQLStr = @spexecutesQLStr + N' OPTION(RECOMPILE)';

BEGIN TRY
EXEC [sp_executesql] @spexecutesQLStr
, N'@StartDate date, @EndDate date'
, @StartDate = @StartDate
, @EndDate = @EndDate;
END TRY
BEGIN CATCH
-- Call procedure to print error information.

EXECUTE dbo.uspPrintError;

-- Roll back any active or uncommittable transactions before
-- inserting information in the ErrorLog.

IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END
GO

```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIColumnDetails]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@TableName	varchar(256)	256

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20141124
-- Description: stored procedure to list column details for each column in a table
-- =====

CREATE PROCEDURE [dbo].[usp_SIColumnDetails]
    -- Add the parameters for the stored procedure here

    @TableName varchar(256) = NULL
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    -- Insert statements for procedure here

    SELECT 'Table' = @TableName,
    c.name 'Column Name',
    t.Name 'Data type',
    c.max_length 'Max Length',
    c.precision ,
    c.scale ,
    c.is_nullable,
```

```
ISNULL(i.is_primary_key, 0) 'Primary Key'  
FROM  
    sys.columns c  
INNER JOIN  
    sys.types t ON c.user_type_id = t.user_type_id  
LEFT OUTER JOIN  
    sys.index_columns ic ON ic.object_id = c.object_id AND ic.column_id =  
c.column_id  
LEFT OUTER JOIN  
    sys.indexes i ON ic.object_id = i.object_id AND ic.index_id = i.index_id  
WHERE  
    c.object_id = OBJECT_ID(@TableName)  
END  
GO
```

[dbo].[usp_SlCountAdults]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@MinAge	tinyint	1
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150610
-- Description: User defined stored procedure to
--               count adults visiting during
--               reporting period
-- =====

CREATE PROCEDURE [dbo].[usp_SlCountAdults]
    -- Add the parameters for the stored procedure here

    @StartDate date = NULL,
    @EndDate date = NULL,
    @MinAge tinyint = 17,
    @DEBUG bit = 0
AS
BEGIN
```

Author: liam

```

-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int, @ReturnError int;

BEGIN TRY

    IF (@StartDate IS NULL)
        SET @StartDate = '18000101'

    IF (@EndDate IS NULL)
        SET @EndDate = GetDate();

    -- Create temporary table

    CREATE Table #TempPotentialAdults
    (
        PersonID int
        , TestID int
        , AgeAtVisit tinyint
        , MostRecentVisit date
        , Birthdate date
        , Visits tinyint
    )

    -- insert values from bloodtest results

    insert Into #TempPotentialAdults (PersonID, MostRecentVisit, TestID)
        select PersonID,MostRecentVisit = SampleDate, TestID = BloodTestResults-
ID
        from BloodtestResults
        where SampleDate >= @StartDate AND SampleDate < @EndDate

    -- insert values from questionnaire

    insert Into #TempPotentialAdults (PersonID, MostRecentVisit, TestID)
        Select PersonID,MostRecentVisit = QuestionnaireDate, TestID =
QuestionnaireID
        from Questionnaire
        where QuestionnaireDate >= @StartDate AND QuestionnaireDate < @End-
Date
        and (ISNULL(Questionnaire.NursingMother,0) = 0 OR
ISNULL(Questionnaire.Pregnant,0) = 0 )

    -- populate birthdate only if the difference from most recent visit to
birthdate is at least minAge

    update #TempPotentialAdults set BirthDate = Person.Birthdate,
        AgeAtVisit = [dbo].[udf_CalculateAge]([Person].[BirthDate],MostRecent-
Visit)
        FROM #TempPotentialAdults
        JOIN Person on Person.PersonID = #TempPotentialAdults.PersonID
        where Datediff(yy,Person.BirthDate,MostRecentVisit) > @MinAge

```

```
    Select AdultsTested = count(distinct PersonID) from #TempPotentialAdults
    where AgeAtVisit > @MinAge

    drop table #TempPotentialAdults
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Add error information to errorlog

    EXECUTE dbo.uspLogError @ErrorLogID = @ErrorLogID OUTPUT;
    SELECT @ReturnError = ERROR_NUMBER();

    -- DROP TABLE ##ReturnedValues;

    RETURN @ReturnError
END CATCH;
END

GO
```

Uses

[dbo].[BloodTestResults]
[dbo].[Person]
[dbo].[Questionnaire]
[dbo].[uspLogError]
[dbo].[uspPrintError]
[dbo].[udf_CalculateAge]



[dbo].[usp_SlCountBloodLeadLevels]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@MinLeadValue	numeric(4,1)	5
@MaxLeadValue	numeric(4,1)	5
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150601
-- Description:  procedure returns the number of
--               entries in the persons table
--               with blood test results within
--               the specified date range, and
--               >= 5 and < 10
-- =====
CREATE PROCEDURE [dbo].[usp_SlCountBloodLeadLevels]
    -- Add the parameters for the stored procedure here
    @StartDate date = NULL,
    @EndDate date = NULL,
    @MinLeadValue numeric(4,1) = NULL,
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SICountBloodLead-Levels

```
@MaxLeadValue Numeric(4,1) = NULL,
@DEBUG bit = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @spexecutesqlStr NVARCHAR(4000), @Recompile BIT = 1, @ErrorLogID int;

    BEGIN TRY
        SELECT @spexecutesqlStr = 'SELECT EBLLTests = count([BloodTestResultsID]) from
[BloodTestResults]
where 1 = 1'

        IF (@MinLeadValue IS NOT NULL)
            SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND Leadvalue >= @MinLead-
Value'

        IF (@MaxLeadValue IS NOT NULL)
            SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND LeadValue < @MaxLead-
Value'

        IF (@StartDate IS NOT NULL)
            SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND SampleDate >= @Start-
Date'

        IF (@EndDate IS NOT NULL)
            SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND SampleDate < @EndDate'

        IF (@Recompile = 1)
            SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

        IF (@DEBUG = 1)
            SELECT @spexecutesqlStr, [StartDate] = @StartDate, [EndDate] = @EndDate,
[MinLeadValue] = @MinLeadValue, [MaxLeadValue] = @MaxLeadValue

            EXEC [sp_executesql] @spexecutesqlStr
                , N'@StartDate date, @EndDate date, @MinLeadValue numeric(4,1), @MaxLeadValue
numeric(4,1)'
                , @StartDate = @StartDate
                , @EndDate = @EndDate
                , @MinLeadValue = @MinLeadValue
                , @MaxLeadValue = @MaxLeadValue

    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;
    END CATCH

```

Author: liam

```
-- Roll back any active or uncommittable transactions before
-- inserting information in the ErrorLog.

IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SICountBloodTests]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@MinLeadValue	numeric(4,1)	5
@MaxLeadValue	numeric(4,1)	5
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150605
-- Description:    procedure returns the number of
--                  blood tests conducted within
--                  the specified date range.
-- =====

CREATE PROCEDURE [dbo].[usp_SICountBloodTests]
    -- Add the parameters for the stored procedure here

    @StartDate date = NULL,
    @EndDate date = NULL,
    @MinLeadValue numeric(4,1) = NULL,
    @MaxLeadValue Numeric(4,1) = NULL,
    @DEBUG bit = 0
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SICountBlood-Tests

```
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @spexecutesqlStr NVARCHAR(4000), @Recompile BIT = 1, @ErrorLogID int;

    BEGIN TRY
        SELECT @spexecutesqlStr = 'SELECT BloodTests = count([BloodTestResultsID]) from
[BloodTestResults]
    where 1 = 1'

        IF (@MinLeadValue IS NOT NULL)
            SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND Leadvalue >= @MinLead-
Value'

        IF (@MaxLeadValue IS NOT NULL)
            SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND LeadValue < @MaxLead-
Value'

        IF (@StartDate IS NOT NULL)
            SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND SampleDate >= @Start-
Date'

        IF (@EndDate IS NOT NULL)
            SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND SampleDate < @EndDate'

        IF ( (DATEDIFF(YYYY,@StartDate,@EndDate)) > 5)
            SET @Recompile = 0

        IF ( (@MaxLeadValue - @MinLeadValue) > 5)
            SET @Recompile = 0

        IF (@Recompile = 1)
            SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

        IF (@DEBUG = 1)
            SELECT @spexecutesqlStr, [StartDate] = @StartDate, [EndDate] = @EndDate,
[MinLeadValue] = @MinLeadValue, [MaxLeadValue] = @MaxLeadValue

            EXEC [sp_executesql] @spexecutesqlStr
                , N'@StartDate date, @EndDate date, @MinLeadValue numeric(4,1), @MaxLeadValue
numeric(4,1)'
                , @StartDate = @StartDate
                , @EndDate = @EndDate
                , @MinLeadValue = @MinLeadValue
                , @MaxLeadValue = @MaxLeadValue

    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.
```

Author: liam

```
EXECUTE dbo.uspPrintError;

-- Roll back any active or uncommittable transactions before

-- inserting information in the ErrorLog.

IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SICountClients]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150605
-- Description: procedure returns the number of
--               blood tests conducted within
--               the specified date range.
-- =====

CREATE PROCEDURE [dbo].[usp_SICountClients]
    -- Add the parameters for the stored procedure here

    @StartDate date = NULL,
    @EndDate date = NULL,
    @DEBUG bit = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```

SET NOCOUNT ON;

DECLARE @spexecutesqlStr NVARCHAR(4000), @Recompile BIT = 1, @ErrorLogID int;

BEGIN TRY

    IF (@StartDate IS NULL)
        SET @StartDate = '18000101';

    IF (@EndDate IS NULL)
        SET @EndDate = GETDATE();

    IF (@StartDate >= @EndDate)
        BEGIN
            DECLARE @ErrorMessage VARCHAR(3000);
            SET @ErrorMessage ='EndDate must be after StartDate: StartDate: ' +
cast(@StartDate as varchar) + ' EndDate: ' + cast(@EndDate as varchar)
            RAISERROR (@ErrorMessage, 11, -1);
            RETURN;
        END

    SELECT @spexecutesqlStr = 'Select Clients = count(PersonID) from (
        SELECT PersonID from bloodTestResults WHERE 1=1 AND SampleDate >= @Start-
Date AND SampleDate < @EndDate
        UNION
        SELECT PersonID from Questionnaire WHERE 1=1 AND QuestionnaireDate >=
@StartDate AND QuestionnaireDate < @EndDate
    ) total'

    IF (@Recompile = 1)
        SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

    IF (@DEBUG = 1)
        SELECT @spexecutesqlStr, [StartDate] = @StartDate, [EndDate] = @EndDate

    EXEC [sp_executesql] @spexecutesqlStr
    , N'@StartDate date, @EndDate date'
    , @StartDate = @StartDate
    , @EndDate = @EndDate

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

```

```
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SlCountFamilyMembers]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@FamilyID	int	4
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20141125
-- Description: stored procedure to count family members
-- =====
CREATE PROCEDURE [dbo].[usp_SlCountFamilyMembers]
    -- Add the parameters for the stored procedure here
    @FamilyID int = NULL,
    @DEBUG BIT = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @spexecutesQLStr NVARCHAR(4000)
        , @Recompile BIT = 1, @ErrorLogID int;
    --IF (@FamilyID IS NULL)

    --BEGIN
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SICountFamily-Members

```
--      RAISERROR ('You must supply at least one parameter.', 11, -1);

--      RETURN;

--END;

SELECT @spexecutesQLStr =
N'SELECT [f].[familyid], FamilyName = [f].[lastname], Members =
count([P].[Lastname]) from [Family] AS [F]
LEFT OUTER JOIN [persontoFamily] [p2f] on [F].[FamilyID] = [p2F].[Familyid]
LEFT OUTER JOIN [Person] AS [P] on [P].[Personid] = [p2f].[Personid]
where 1=1';

IF (@FamilyID IS NOT NULL)
SELECT @spexecutesQLStr = @spexecutesQLStr
+ N' AND [f].[familyID] = @Family_ID';

SELECT @spexecutesQLStr = @spexecutesQLStr
+ N' group by [f].[familyid],[f].[lastname]
order by [f].[lastname],[f].[familyid]';

IF (@FamilyID IS NULL)
SET @Recompile = 0;

IF @Recompile = 1
SELECT @spexecutesQLStr = @spexecutesQLStr + N' OPTION(RECOMPILE)';

BEGIN TRY
IF (@DEBUG = 1)
SELECT @spexecutesQLStr, 'FamilyID' = @FamilyID;

EXEC [sp_executesql] @spexecutesQLStr
, N'@Family_ID int'
, @Family_ID = @FamilyID;
END TRY
BEGIN CATCH
-- Call procedure to print error information.

EXECUTE dbo.uspPrintError;

-- Roll back any active or uncommittable transactions before
-- inserting information in the ErrorLog.

IF XACT_STATE() <> 0
BEGIN
ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
```

Author: liam

```
    END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SlCountHomeVisitSoilSample]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150611
-- Description: User defined stored procedure to
--               count clients that have a status
--               of home visit and/or soil sample
--               during the reporting period
-- =====

CREATE PROCEDURE [dbo].[usp_SlCountHomeVisitSoilSample]
    -- Add the parameters for the stored procedure here

    @StartDate date = NULL,
    @EndDate date = NULL,
    @DEBUG bit = 0
AS
BEGIN
```

Author: liam

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SICountHomeVisit-SoilSample

```
-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @spexecutesqlStr NVARCHAR (4000),
@Recompile BIT = 1, @ErrorLogID int, @ReturnError int;

IF (@StartDate IS NULL)
    SET @StartDate = '18000101'

IF (@ENDDate IS NULL)
    SET @EndDate = GetDate();

select @spexecutesqlStr ='select [HomeVisitSoilSamples] = count(PersonID) from (
SELECT PersonID
from BloodTestResults where SampleDate >= @StartDate and Sample-
Date < @EndDate
                AND ClientStatusID in (    SELECT [TS].[StatusID] from
[TargetStatus] AS [TS]
                                where TargetType = ''Person''
                                AND StatusName in (''Home
visit'', ''Home Visit and Soil Sample'', ''Soil Sample'')
)
                UNION
-- people with questionnaire but no blood test during reporting
period
Select Q.PersonID
from Questionnaire AS Q
LEFT OUTER JOIN [BloodTestResults] AS [BTR] on
[BTR].[BloodTestResultsID] = (
                                select top 1 [BloodTest-
ResultsID] from [BloodTestResults]
                                where [BloodTest-
Results].[PersonID] = [Q].[PersonID]
                                -- AND SampleDate >=
@StartDate AND SampleDate < @EndDate
                                AND BTR.ClientStatusID
                                in (    SELECT
[TS].[StatusID] from [TargetStatus] AS [TS]
                                where
TargetType = ''Person''
                                AND Status-
Name in (''Home visit'', ''Home Visit and Soil Sample'', ''Soil Sample'')
)
                                order by SampleDate
desc
)
                where QuestionnaireDate >= @StartDate and Questionnaire-
Date < @EndDate
                AND BTR.ClientStatusID
                in (    SELECT [TS].[StatusID] from [TargetStatus]
AS [TS]
                                where TargetType = ''Person''
                                AND StatusName in (''Home visit'', ''Home
Visit and Soil Sample'', ''Soil Sample'')
)
```

Author: liam

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SICountHomeVisit-SoilSample

```
        )
    ) HomeVisitSoilSamples'

IF @Recompile = 1
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesqlStr, 'StartDate' = @StartDate, 'EndDate' = @EndDate,
'DEBUG' = @Debug

        EXEC [sp_executesql] @spexecutesqlStr
        , N'@StartDate datetime, @EndDate datetime'
        , @StartDate = @StartDate
        , @EndDate = @EndDate;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Add error information to errorlog

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    SELECT @ReturnError = ERROR_NUMBER();

    -- DROP TABLE ##ReturnedValues;

    RETURN @ReturnError
END CATCH;
END

GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SICountNewClients]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150605
-- Description: procedure returns the number of
--               clients onboarded during the
--               reporting period.
-- =====

CREATE PROCEDURE [dbo].[usp_SICountNewClients]
    -- Add the parameters for the stored procedure here

    @StartDate date = NULL,
    @EndDate date = NULL,
    @DEBUG bit = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @spexecutesqlStr NVARCHAR(4000), @Recompile BIT = 1, @ErrorLogID int;

BEGIN TRY
    IF (@StartDate IS NULL)
        SET @StartDate = '18000101';

    IF (@EndDate IS NULL)
        SET @EndDate = GETDATE();

    IF (@StartDate >= @EndDate)
        BEGIN
            DECLARE @ErrorString VARCHAR(3000);
            SET @ErrorString ='EndDate must be after StartDate: StartDate: ' +
cast(@StartDate as varchar) + ' EndDate: ' + cast(@EndDate as varchar)
            RAISERROR (@ErrorString, 11, -1);
            RETURN;
        END

    SELECT @spexecutesqlStr = 'Select NewClients = count(PersonID) from Person
WHERE isClient = 1'

    IF (@StartDate IS NOT NULL)
        SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND CreatedDate >= @Start-
Date'

    IF (@EndDate IS NOT NULL)
        SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND CreatedDate < @EndDate'

    IF (@Recompile = 1)
        SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

    IF (@DEBUG = 1)
        SELECT @spexecutesqlStr, [StartDate] = @StartDate, [EndDate] = @EndDate

    EXEC [sp_executesql] @spexecutesqlStr
    , N'@StartDate date, @EndDate date'
    , @StartDate = @StartDate
    , @EndDate = @EndDate

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
        BEGIN
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SICountNew-Clients

```
ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SlCountNewPeople]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150605
-- Description: procedure returns the number of
--               blood tests conducted within
--               the specified date range.
-- =====

CREATE PROCEDURE [dbo].[usp_SlCountNewPeople]
    -- Add the parameters for the stored procedure here

    @StartDate date = NULL,
    @EndDate date = NULL,
    @DEBUG bit = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @spexecutesqlStr NVARCHAR(4000), @Recompile BIT = 1, @ErrorLogID int;

BEGIN TRY

    IF (@StartDate IS NULL)
        SET @StartDate = '18000101';

    IF (@EndDate IS NULL)
        SET @EndDate = GETDATE();

    IF (@StartDate >= @EndDate)
        BEGIN
            DECLARE @ErrorMessage VARCHAR(3000);
            SET @ErrorMessage ='EndDate must be after StartDate: StartDate: ' +
cast(@StartDate as varchar) + ' EndDate: ' + cast(@EndDate as varchar)
            RAISERROR (@ErrorMessage, 11, -1);
            RETURN;
        END

    SELECT @spexecutesqlStr = 'Select NewPeople = count(PersonID) from Person
WHERE 1=1'

    IF (@StartDate IS NOT NULL)
        SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND CreatedDate >= @Start-
Date'

    IF (@EndDate IS NOT NULL)
        SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND CreatedDate < @EndDate'

    IF (@Recompile = 1)
        SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

    IF (@DEBUG = 1)
        SELECT @spexecutesqlStr, [StartDate] = @StartDate, [EndDate] = @EndDate

        EXEC [sp_executesql] @spexecutesqlStr
        , N'@StartDate date, @EndDate date'
        , @StartDate = @StartDate
        , @EndDate = @EndDate

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SICountNew-People

```
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SICountNursingInfants]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150607
-- Description: procedure returns the number of
--               nursing infants that either had a
--               bloodtest or completed a questionnaire
--               within the specified date range.
-- =====
CREATE PROCEDURE [dbo].[usp_SICountNursingInfants]
    -- Add the parameters for the stored procedure here

    @StartDate date = NULL,
    @EndDate date = NULL,
    @DEBUG bit = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @spexecutesqlStr NVARCHAR(4000), @Recompile BIT = 1, @ErrorLogID int;

BEGIN TRY

    IF (@StartDate IS NULL)
        SET @StartDate = '18000101';

    IF (@EndDate IS NULL)
        SET @EndDate = GETDATE();

    IF (@StartDate >= @EndDate)
        BEGIN
            DECLARE @ErrorMessage VARCHAR(3000);
            SET @ErrorMessage ='EndDate must be after StartDate: StartDate: ' +
cast(@StartDate as varchar) + ' EndDate: ' + cast(@EndDate as varchar)
            RAISERROR (@ErrorMessage, 11, -1);
            RETURN;
        END

    SELECT @spexecutesqlStr = 'Select [NursingInfants] = COUNT(PersonID) from (
                                Select BTR.PersonID,Q.NursingInfant from BloodTest-
Results AS BTR
                                LEFT OUTER JOIN [Questionnaire] AS [Q] on
[Q].[QuestionnaireID] = (
                                select TOP 1
[QuestionnaireID] from [Questionnaire]
                                where
[Questionnaire].[PersonID] = [BTR].[PersonID]
                                AND Questionnaire-
Date >= @StartDate AND QuestionnaireDate < @EndDate
                                order by Nursing-
Infant desc
)
                                where SampleDate >= @StartDate and SampleDate <
@EndDate AND Q.NursingInfant = 1

                                UNION
                                SELECT PersonID,NursingInfant from Questionnaire
where QuestionnaireDate >= @StartDate and QuestionnaireDate < @EndDate
                                AND NursingInfant = 1
)
ClientsinReportingPeriod
where ClientsinReportingPeriod.NursingInfant = 1'

    IF ((DateDiff(YYYY,@StartDate,@EndDate) > 5))
        SET @Recompile = 0;

    IF (@Recompile = 1)
        SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

    IF (@DEBUG = 1)
```

```
SELECT @spexecutesqlStr, [StartDate] = @StartDate, [EndDate] = @EndDate

EXEC [sp_executesql] @spexecutesqlStr
, N'@StartDate date, @EndDate date'
, @StartDate = @StartDate
, @EndDate = @EndDate

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SICountNursingMothers]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150607
-- Description: procedure returns the number of
--               nursing Mothers that either had a
--               bloodtest or completed a questionnaire
--               within the specified date range.
-- =====
CREATE PROCEDURE [dbo].[usp_SICountNursingMothers]
    -- Add the parameters for the stored procedure here

    @StartDate date = NULL,
    @EndDate date = NULL,
    @DEBUG bit = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @spexecutesqlStr NVARCHAR(4000), @Recompile BIT = 1, @ErrorLogID int;

BEGIN TRY

    IF (@StartDate IS NULL)
        SET @StartDate = '18000101';

    IF (@EndDate IS NULL)
        SET @EndDate = GETDATE();

    IF (@StartDate >= @EndDate)
        BEGIN
            DECLARE @ErrorMessage VARCHAR(3000);
            SET @ErrorMessage ='EndDate must be after StartDate: StartDate: ' +
cast(@StartDate as varchar) + ' EndDate: ' + cast(@EndDate as varchar)
            RAISERROR (@ErrorMessage, 11, -1);
            RETURN;
        END

    SELECT @spexecutesqlStr = 'Select [NursingMothers] = COUNT(PersonID) from (
                                Select BTR.PersonID,Q.NursingMother from BloodTest-
Results AS BTR
                                LEFT OUTER JOIN [Questionnaire] AS [Q] on
[Q].[QuestionnaireID] = (
                                select top 1
[QuestionnaireID] from [Questionnaire]
                                where
[Questionnaire].[PersonID] = [BTR].[PersonID]
                                AND QuestionnaireDate
>= @StartDate AND QuestionnaireDate < @EndDate
                                order by NursingMother
desc
)
                                where SampleDate >= @StartDate and SampleDate < @End-
Date AND Q.NursingMother = 1

                                UNION
                                SELECT PersonID,NursingMother from Questionnaire where
QuestionnaireDate >= @StartDate and QuestionnaireDate < @EndDate
                                AND NursingMother = 1
                                ) ClientsinReportingPeriod
                                where ClientsinReportingPeriod.NursingMother = 1'

    IF ((DateDiff(YYYY,@StartDate,@EndDate) > 5))
        SET @Recompile = 0;

    IF (@Recompile = 1)
        SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

    IF (@DEBUG = 1)
```

```
SELECT @spexecutesqlStr, [StartDate] = @StartDate, [EndDate] = @EndDate

EXEC [sp_executesql] @spexecutesqlStr
, N'@StartDate date, @EndDate date'
, @StartDate = @StartDate
, @EndDate = @EndDate

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SlCountPeople]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Max_Age	int	4
@StartDate	date	3
@EndDate	date	3

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 2/13/2014
-- Description: procedure returns the number of entries in the persons table, being
-- the number of participants
-- =====

CREATE PROCEDURE [dbo].[usp_SlCountPeople]
    -- Add the parameters for the stored procedure here

    @Max_Age int = NULL,
    @StartDate date = NULL,
    @EndDate date = NULL

    AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @spexecutesqlStr NVARCHAR(4000), @Recompile BIT = 1, @ErrorLogID int, @Max-
Age int;
```

```

BEGIN TRY
    SELECT @spexecutesqlStr = 'SELECT NewPeople = count([PersonId]) from [person]
WHERE 1=1'

    IF (@Max_Age IS NOT NULL)
        SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND [Age] <= @MaxAge';

    IF (@StartDate IS NOT NULL)
        SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND CreatedDate >= @Start-
Date';

    IF (@EndDate IS NOT NULL)
        SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND CreatedDate < @EndDate';

    IF @Recompile = 1
        SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

    EXEC [sp_executesql] @spexecutesqlStr
    , N'@MaxAge VARCHAR(50),@StartDate date, @EndDate date'
    , @MaxAge = @Max_Age
    , @StartDate = @StartDate
    , @EndDate = @EndDate

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO

```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SlCountPeopleByAge]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

SQL Script

```
-- =====

-- Author:      Liam Thier

-- Create date: 20141222

-- Description:    returns count of people grouped by

--                  age. If a lastname is passed in

--                  displays a list of people with that lastname

-- =====

CREATE PROCEDURE [dbo].[usp_SlCountPeopleByAge]
    -- Add the parameters for the stored procedure here

    -- @Last_Name varchar(50) = NULL

    AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from

    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @spexecutesqlStr      NVARCHAR (4000),
            @Recompile   BIT = 1;

    -- Insert statements for procedure here

    select @spexecutesqlStr ='select Age, ''Personcount'' = count(PersonID)
                                from [person]
                                where isClient = 1'
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SICountPeopleByAge

```
-- Return all families and associated properties if nothing was passed in

--IF (@LastName IS NOT NULL)

    -- SELECT @spexecutesqlStr = @spexecutesqlStr + ' and [LastName] = @LastName'

--ELSE

    -- SET @Recompile = 0

-- group people by age

SELECT
@spexecutesqlStr = @spexecutesqlStr + ' group by [dbo].udf_CalculateAge(Birth-
Date,GetDate())'

-- order by age

SELECT
@spexecutesqlStr = @spexecutesqlStr + ' order by Age'

IF @Recompile = 1
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

EXEC [sp_executesql] @spexecutesqlStr
--, N'@LastName varchar(50)'

--, @LastName = @LastName;

END
GO
```

Author: liam

[dbo].[usp_SICountPeopleByAgeGroup]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150112
-- Description:    returns count of people grouped by
--                  age categories. If a lastname is passed in
--                  displays a list of people with that lastname
-- =====
CREATE PROCEDURE [dbo].[usp_SICountPeopleByAgeGroup]
    -- Add the parameters for the stored procedure here
    -- @Last_Name varchar(50) = NULL
    @DEBUG BIT = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @spexecutesqlStr NVARCHAR (4000),
            @Recompile BIT = 1;
```

Author: liam

```
; with AgeGroups as
(
    SELECT CASE
        WHEN Age < 1 THEN '0'
        WHEN Age >= 1 and Age < 4 THEN '01 - 03'
        WHEN Age >= 4 AND Age < 7 THEN '04 - 06'
        WHEN Age >= 7 AND Age < 18 THEN '07 - 17'
        ELSE '18 and Over'
    END AS Groups
    -- , MaxAge = max(Person.Age)

    FROM Person
    where isClient = 1
)

SELECT ROW_NUMBER() OVER(ORDER BY Groups DESC) AS Row, AgeGroups =
Coalesce(Groups,'Total'),
Clients = Count(Groups) From AgeGroups group by
Groups-- with ROLLUP

-- Insert statements for procedure here

select @spexecutesqlStr ='SELECT AgeGroups = Coalesce(Groups,''Total''),
Clients = Count(Groups) From AgeGroups group by Groups
with ROLLUP'
END
GO
```

Uses

[dbo].[Person]



Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Last_Name	varchar(50)	50

SQL Script

```
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====

CREATE PROCEDURE [dbo].[usp_SICountPeopleByLastName]
    @Last_Name VARCHAR(50) = NULL
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int, @spExecutesqlStr NVARCHAR(4000), @Recompile BIT = 1;

    BEGIN TRY
        SELECT @spexecutesqlStr = 'SELECT [lastname], ''Members'' = count([firstname])
from [person] WHERE 1=1';

        if (@Last_Name is not NULL)
        BEGIN
            SET @Recompile = 1;
            SELECT @spExecutesqlStr = @spExecutesqlStr + ' AND [person].[LastName] =
@LastName'
        END
    END TRY
    BEGIN CATCH
        -- Error handling logic here
    END CATCH
END
```

Author: liam

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SICountPeopleBy-
LastName

```
END
ELSE
    SET @Recompile = 0

    -- Group by last name for counting purposes

    SELECT @spExecutesqlStr = @spExecutesqlStr + ' group by [lastname]'

    -- force recompile for selective query

    IF @Recompile = 1
        SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

    EXEC [sp_executesql] @spExecutesqlStr
    , N'@LastName VARCHAR(50)'
    , @LastName = @Last_Name;

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before

    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SlCountPregnantWomen]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150610
-- Description: User defined stored procedure to
--               count Pregnant Women
-- =====

CREATE PROCEDURE [dbo].[usp_SlCountPregnantWomen]
    -- Add the parameters for the stored procedure here

    @StartDate date = NULL,
    @EndDate date = NULL,
    @DEBUG bit = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
```

```
SET NOCOUNT ON;

DECLARE @spexecutesqlStr NVARCHAR (4000),
        @Recompile BIT = 1, @ErrorLogID int, @ReturnError int;

IF (@StartDate IS NULL)
    SET @StartDate = '18000101'

IF (@ENDDate IS NULL)
    SET @EndDate = GetDate();

select @spexecutesqlStr ='Select [PregnantWomen] = COUNT(PersonID) from (
                                Select BTR.PersonID,Q.Pregnant from BloodTestResults AS
BTR
                                LEFT OUTER JOIN [Questionnaire] AS [Q] on
[Q].[QuestionnaireID] =
                                select top 1
[QuestionnaireID] from [Questionnaire]
                                where
[Questionnaire].[PersonID] = [BTR].[PersonID]
                                AND QuestionnaireDate
>= @StartDate AND QuestionnaireDate < @EndDate
                                order by Pregnant desc
)
                                where SampleDate >= @StartDate and SampleDate < @End-
Date AND Q.Pregnant = 1

                                UNION
                                SELECT PersonID,Pregnant from Questionnaire where
QuestionnaireDate >= @StartDate and QuestionnaireDate < @EndDate
                                AND Pregnant = 1
                                ) ClientsinReportingPeriod
                                where ClientsinReportingPeriod.Pregnant = 1'

IF @Recompile = 1
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesqlStr, 'StartDate' = @StartDate, 'ENDDate' = @EndDate,
'DEBUG' = @Debug

        EXEC [sp_executesql] @spexecutesqlStr
        , N'@StartDate datetime, @EndDate datetime'
        , @StartDate = @StartDate
        , @EndDate = @EndDate;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Add error information to errorlog

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SICountPregnant-Women

```
SELECT @ReturnError = ERROR_NUMBER();

-- DROP TABLE ##ReturnedValues;

RETURN @ReturnError
END CATCH;
END

GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SlDaycare]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150319
-- Description: returns daycare name, id, description
-- =====

CREATE PROCEDURE [dbo].[usp_SlDaycare]
    -- Add the parameters for the stored procedure here

    @DEBUG BIT = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    -- Insert statements for procedure here

    select DaycareID,DaycareName,DaycareDescription from Daycare order by DaycareName
END
GO
```

Uses

[dbo].[Daycare]



[dbo].[usp_SIEditBloodTestResultsWebScreenInformation]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150618
-- Description: stored procedure to select
--               bloodtestresults edit screen info
-- =====

CREATE PROCEDURE [dbo].[usp_SIEditBloodTestResultsWebScreenInformation]
    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @DEBUG BIT = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @spexecutesQLStr NVARCHAR(4000)
        , @Recompile BIT = 1, @ErrorLogID int;
```

```
IF (@PersonID IS NULL)
BEGIN
    RAISERROR ('You must supply a person.', 11, -1);
    RETURN;
END;

SELECT @spexecutesQLStr =
N'select [BTR].[BloodTestResultsID]
,[BTR].[SampleDate]
,[BTR].[LabSubmissiondate]
,[L].[LabName]
,[BTR].[LeadValue]
,[FollowupDate] = [P].[RetestDate]
,[ST].[SampleTypeName]
,[TS].[StatusName]
,[BTR].[HemoglobinValue]
from [BloodTestResults] AS [BTR]
LEFT OUTER JOIN [Person] AS [P] on [BTR].[PersonID] = [P].[PersonID]
LEFT OUTER JOIN [Lab] AS [L] on [BTR].[LabID] = [L].[LabID]
LEFT OUTER JOIN [SampleType] AS [ST] on [BTR].[SampleTypeID] = [ST].[SampleTypeID]
LEFT OUTER JOIN [TargetStatus] AS [TS] on [BTR].[ClientStatusID] = [TS].[StatusID]
where [BTR].[PersonID] = @PersonID';

SELECT @spexecutesQLStr = @spexecutesQLStr
+ N' order by [BTR].[SampleDate] desc';

IF @Recompile = 1
    SELECT @spexecutesQLStr = @spexecutesQLStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesQLStr, 'PersonID' = @PersonID;

    EXEC [sp_executesql] @spexecutesQLStr
        , N'@PersonID int'
        , @PersonID = @PersonID;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

```

```
EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]



[dbo].[usp_SIEditClientInfoWebScreenInformation]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150408
-- Description: stored procedure to select
--               person edit screen info
-- =====

CREATE PROCEDURE [dbo].[usp_SIEditClientInfoWebScreenInformation]
    -- Add the parameters for the stored procedure here
    @PersonID int = NULL,
    @DEBUG BIT = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @spexecutesQLStr NVARCHAR(4000)
        , @Recompile BIT = 1, @ErrorLogID int;
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SIEditClientInfo-WebScreenInformation

```
IF (@PersonID IS NULL)
BEGIN
    RAISERROR ('You must supply a person.', 11, -1);
    RETURN;
END;

SELECT @spexecutesQLStr =
N'select [P].[PersonID],[P].[LastName],[P].[FirstName],[P].[MiddleName]
,[P].[Birthdate],[P].[Gender]
,[P].[isClient]
,[L].[LanguageID]
,[L].[LanguageName]
,[E].[EthnicityID]
,[E].[Ethnicity]
,[P].[Moved]
,[MovedOutOfCounty] = cast([P].[OutofSite] as varchar)
,[TravelV] = cast([P].[ForeignTravel] as varchar)
,[P].[EmailAddress]
from [Person] AS [P]
LEFT OUTER JOIN [PersontoLanguage] AS [PL] on [P].[PersonID] = [PL].[PersonID]
LEFT OUTER JOIN [Language] AS [L] ON [PL].[LanguageID] = [L].[LanguageID]
LEFT OUTER JOIN [PersontoEthnicity] AS [PE] ON [PE].[PersonID] = [P].[PersonID]
LEFT OUTER JOIN [Ethnicity] AS [E] ON [PE].[EthnicityID] = [E].[EthnicityID]
where [P].[PersonID] = @PersonID';

IF EXISTS ( SELECT PersonID from PersontoLanguage where PersonID = @PersonID )
SELECT @spexecutesQLStr = @spexecutesQLStr
+ N' AND [PL].[isPrimaryLanguage] = 1';

SELECT @spexecutesQLStr = @spexecutesQLStr
+ N' order by [L].[CreatedDate] desc';

IF @Recompile = 1
SELECT @spexecutesQLStr = @spexecutesQLStr + N' OPTION(RECOMPILE)';

BEGIN TRY
IF (@DEBUG = 1)
SELECT @spexecutesQLStr, 'PersonID' = @PersonID;

EXEC [sp_executesql] @spexecutesQLStr
, N'@PersonID int'
, @PersonID = @PersonID;
END TRY
BEGIN CATCH
-- Call procedure to print error information.

EXECUTE dbo.uspPrintError;

-- Roll back any active or uncommittable transactions before
-- inserting information in the ErrorLog.
```

Author: liam

```
IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[PersontoLanguage]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIEditFamilyWebScreenInformation]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Family_ID	int	4
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150405
-- Description: returns Family Lastname, Primary Address,
--               Primary phononenumber, Secondary phononenumber,
--               number of smokers, number of pets,
--               if pets are in and out pets,
--               if pets are washed frequently
-- =====
CREATE PROCEDURE [dbo].[usp_SIEditFamilyWebScreenInformation]
    -- Add the parameters for the stored procedure here
    @Family_ID INT = NULL,
    @DEBUG BIT = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

DECLARE @PrimaryPhoneNumber bigint, @SecondaryPhoneNumber bigint,
        @spexecutesQLStr NVARCHAR(4000), @Recompile BIT = 1, @ErrorLogID int;

IF (@Family_ID IS NULL)
BEGIN
    RAISERROR ('You must supply the Family.', 11, -1);
    RETURN;
END;

-- Select Primary Phone number

select @PrimaryPhoneNumber = dbo.udf_SlFamilyPhoneNumber(@Family_ID, 1)

-- Select Secondary Phone number

select @SecondaryPhoneNumber = dbo.udf_SlFamilyPhoneNumber(@Family_ID, 2)

SELECT @spexecutesQLStr =
N'SELECT [F].[FamilyID],[F].[Lastname],[P].[AddressLine1],[P].[AddressLine2]
    ,[P].[City],[P].[State],[P].[ZipCode],YearBuilt = cast([P].[YearBuilt] as
date)
    ,MoveinDate = cast(StartDate as date), MoveoutDate = cast(EndDate as date)
    ,[OwnerOccupied] = cast([P].[isOwnerOccupied] as varchar)
    , PrimaryPhoneNumber = @PrimaryPhoneNumber, SecondaryPhoneNumber =
@SecondaryPhoneNumber
    ,[F].[NumberofSmokers],[F].[Pets],Petsonandout = cast([F].[Petsonandout] as
varchar)
    ,[P].[OwnerContactInformation]
FROM [Family] AS [F]
JOIN [Property] AS [P] ON [F].[PrimaryPropertyID] = [P].[PropertyID]
JOIN [FamilytoProperty] AS [F2P] ON [F].[FamilyID] = [F2P].[FamilyID] AND
[F].[PrimaryPropertyID] = [F2P].[PropertyID]
WHERE 1 = 1'

IF (@Family_ID IS NULL)
    SET @Recompile = 0;

IF (@Family_ID IS NOT NULL)
    SELECT @spexecutesQLStr = @spexecutesQLStr + ' and [F].[FamilyID] = @FamilyID
ORDER by [F].[FamilyID] desc'

IF @Recompile = 1
    SELECT @spexecutesQLStr = @spexecutesQLStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesQLStr, 'FamilyID' = @Family_ID, 'PrimaryPhoneNumber' =
@PrimaryPhoneNumber, 'SecondaryPhoneNumber' = @SecondaryPhoneNumber;
```

```
EXEC [sp_executesql] @spexecutesQLStr
    , N'@FamilyID int, @PrimaryPhoneNumber bigint, @SecondaryPhoneNumber
bigint'
    , @FamilyID = @Family_ID
    , @PrimaryPhoneNumber = @PrimaryPhoneNumber
    , @SecondaryPhoneNumber = @SecondaryPhoneNumber;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]
[dbo].[udf_SIFamilyPhoneNumber]



[dbo].[usp_SIEditPropertyWebScreenInformation]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Property_ID	int	4
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150405
-- Description:    returns AddressLine1, Addressline2
--                  City, State, and Zipcode
--                  of a specific property
--                  if no property ID is passed in,
--                  informatin is returned for all properties
-- =====
CREATE PROCEDURE [dbo].[usp_SIEditPropertyWebScreenInformation]
    -- Add the parameters for the stored procedure here
    @Property_ID INT = NULL,
    @DEBUG BIT = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

DECLARE @spexecutesQLStr NVARCHAR(4000), @Recompile BIT = 1, @ErrorLogID int;

SELECT @spexecutesQLStr =
N'SELECT [P].[PropertyID], [P].[AddressLine1], [P].[AddressLine2]
, [P].[City], [P].[State], [P].[ZipCode]
FROM [Property] AS [P]
WHERE 1 = 1'

IF (@Property_ID IS NULL)
    SET @Recompile = 0;

IF (@Property_ID IS NOT NULL)
    SELECT @spexecutesQLStr = @spexecutesQLStr + ' and PropertyID = @PropertyID'
ORDER by PropertyID desc'

IF @Recompile = 1
    SELECT @spexecutesQLStr = @spexecutesQLStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesQLStr, 'PropertyID' = @Property_ID;

    EXEC [sp_executesql] @spexecutesQLStr
        , N'@PropertyID int'
        , @PropertyID = @Property_ID;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIEditQuestionnaireWebScreenInformation]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150618
-- Description: stored procedure to select
--               questionnaire edit screen info
-- =====

CREATE PROCEDURE [dbo].[usp_SIEditQuestionnaireWebScreenInformation]
    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @DEBUG BIT = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @spexecutesQLStr NVARCHAR(4000)
        , @Recompile BIT = 1, @ErrorLogID int;
```

```
IF (@PersonID IS NULL)
BEGIN
    RAISERROR ('You must supply a person.', 11, -1);
    RETURN;
END;

SELECT @spexecutesQLStr =
N'select [Q].[QuestionnaireID]
, [Q].[QuestionnaireDate]
, [Q].[isExposedtoPeelingPaint]
, [Q].[PaintDate]
, [Q].[VisitRemodeledProperty]
, [Q].[VisitsOldHomes]
, [Q].[RemodelPropertyDate]
, [Q].[isTakingVitamins]
, [Q].[FrequentHandWashing]
, [Q].[isUsingBottle]
, [Q].[NursingMother]
, [Q].[Pregnant]
, [Q].[NursingInfant]
, [Q].[isUsingPacifier]
, [Q].[BitesNails]
, [Q].[EatOutside]
, [Q].[NonFoodinMouth]
, [Q].[NonFoodEating]
, [Q].[Suckling]
, [Q].[Mouthing]
from [Questionnaire] AS [Q]
where [Q].[PersonID] = @PersonID';

SELECT @spexecutesQLStr = @spexecutesQLStr
+ N' order by [Q].[QuestionnaireDate] desc';

IF @Recompile = 1
    SELECT @spexecutesQLStr = @spexecutesQLStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesQLStr, 'PersonID' = @PersonID;

    EXEC [sp_executesql] @spexecutesQLStr
        , N'@PersonID int'
        , @PersonID = @PersonID;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

```

```
IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIFamilyMembers]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@FamilyID	int	4
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150215
-- Description: stored procedure to list family members
-- =====

CREATE PROCEDURE [dbo].[usp_SIFamilyMembers]
    -- Add the parameters for the stored procedure here

    @FamilyID int = NULL,
    @DEBUG BIT = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @spexecutesQLStr NVARCHAR(4000)
        , @Recompile BIT = 1, @ErrorLogID int;

    --IF (@FamilyID IS NULL)

    --BEGIN
```

```

--      RAISERROR ('You must supply at least one parameter.', 11, -1);

--      RETURN;

--END;

SELECT @spexecutesQLStr =
N'SELECT [f].[familyid], FamilyName = [f].[lastname], [P].[LastName],
[P].[FirstName] from [Family] AS [F]
LEFT OUTER JOIN [persontoFamily] [p2f] on [F].[FamilyID] = [p2F].[Familyid]
LEFT OUTER JOIN [Person] AS [P] on [P].[Personid] = [p2f].[Personid]
where 1=1';

IF (@FamilyID IS NOT NULL)
    SELECT @spexecutesQLStr = @spexecutesQLStr
    + N' AND [f].[familyID] = @Family_ID';

SELECT @spexecutesQLStr = @spexecutesQLStr
+ N' order by [f].[lastname],[f].[familyid]';

IF (@FamilyID IS NULL)
    SET @Recompile = 0;

IF @Recompile = 1
    SELECT @spexecutesQLStr = @spexecutesQLStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesQLStr, 'FamilyID' = @FamilyID;

    EXEC [sp_executesql] @spexecutesQLStr
        , N'@Family_ID int'
        , @Family_ID = @FamilyID;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

```

```
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SlFamilyNametoProperty]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Family_Name	varchar(50)	50

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20141123
-- Description: User defined stored procedure to
--               select family and property address
-- =====
CREATE PROCEDURE [dbo].[usp_SlFamilyNametoProperty]
    -- Add the parameters for the stored procedure here
    @Family_Name varchar(50) = NULL
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @spexecutesqlStr      NVARCHAR (4000),
            @Recompile   BIT = 1, @ErrorLogID int;

    -- Insert statements for procedure here
    select @spexecutesqlStr ='SELECT ''FamilyName'' = [F].[LastName],[Prop].[Street-
```

Author: liam

```
Number], [Prop].[Street], [Prop].[StreetSuffix], [Prop].[ZipCode]
    from [family] AS [F]
    join [Property] as [Prop] on [F].[PrimaryPropertyID] = [Prop].[PropertyID]
    where 1 = 1'

    -- Return all families and associated properties if nothing was passed in

    IF (@Family_Name IS NOT NULL)
        SELECT @spexecutesqlStr = @spexecutesqlStr + ' and [F].[LastName] = @Family-
Name'
    ELSE
        SET @Recompile = 0

    -- order by last name

    SELECT @spexecutesqlStr = @spexecutesqlStr + N' order by [F].[LastName]'

    IF @Recompile = 1
        SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

    BEGIN TRY
        EXEC [sp_executesql] @spexecutesqlStr
        , N'@FamilyName varchar(50)'
        , @FamilyName = @Family_Name;
    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Add error information to errorlog

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIHobby]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150620
-- Description: returns hobby name, id, description
-- =====

CREATE PROCEDURE [dbo].[usp_SIHobby]
    -- Add the parameters for the stored procedure here

    @DEBUG BIT = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    -- Insert statements for procedure here

    select HobbyID,HobbyName,HobbyDescription from Hobby order by HobbyName

END
GO
```

Uses

[dbo].[Hobby]

[dbo].[usp_SLInsertedData]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Last_Name	varchar(50)	50

SQL Script

```

        , [P].[LastName]
        , [P].[MiddleName]
        , [P].[FirstName]
        , [P].[BirthDate]
        , [P].[Gender]
        , ''StreetAddress'' = cast([Prop].[StreetNumber] as
varchar)
                + '' ''+ cast([Prop].[Street] as varchar) + ''
                + cast([Prop].[StreetSuffix] as varchar)
        , [Prop].[ApartmentNumber]
        , [Prop].[City]
        , [Prop].[State]
        , [Prop].[Zipcode]
        , ''PrimaryPhoneNumber'' = [Ph].[PhoneNumber]
        , [L].[LanguageName]
        , [F].[NumberofSmokers]
        , [F].[Pets]
        , [F].[inandout]
        , [F].[Notes]
        , [P].[Moved]
        , [P].[ForeignTravel]
        , [P].[OutofSite]
        , [H].[HobbyName]
        , [P].[Notes]
        , [P].[isSmoker]
        , [P].[RetestDate]
        , [Q].[QuestionnaireDate]
        , [Q].[isExposedtoPeelingPaint]
        , ''PaintAge'' = [Q].[RemodeledPropertyAge]
        , [Q].[VisitRemodeledProperty]
        , ''RemodelPropertyAge'' = [Q].[RemodeledPropertyAge]
        , [Q].[isTakingVitamins]
        , [Q].[FrequentHandWashing]
        , [Q].[isUsingBottle]
        , [Q].[isNursing]
        , [Q].[isUsingPacifier]
        , [Q].[BitesNails]
        , [Q].[EatOutside]
        , [Q].[NonFoodinMouth]
        , [Q].[NonFoodEating]
        , [Q].[Suckling]
        , [Q].[Daycare]
        , [Q].[Source]
        , [Q].[Notes]
        , [BTR].[SampleDate]
        , [BTR].[LabSubmissionDate]
        , [Lab].[LabName]
        , ''What is status code?'''
        , [BTR].[HemoglobinValue]
FROM [LeadTrackingTesting-Liam].[dbo].[Person] AS [P]
LEFT OUTER JOIN [PersonstoFamily] as [P2F] on [P].[PersonID] =
[P2F].[PersonID]
LEFT OUTER JOIN [Family] AS [F] on [F].[FamilyID] =
[P2F].[FamilyID]

```

```

        LEFT OUTER JOIN [PersontoProperty] as [P2P] on [P].[PersonID] =
[P2P].[PersonID]
        LEFT OUTER JOIN [Questionnaire] as [Q] on [P].[PersonID] =
[Q].[PersonID]
        LEFT OUTER JOIN [BloodTestResults] as [BTR] on [P].[PersonID] =
[BTR].[PersonID]
        LEFT OUTER JOIN [PersontoLanguage] as [P2L] on [P2L].[Person-
ID] = [P].[PersonID]
        LEFT OUTER JOIN [Language] as [L] on [L].LanguageID =
[P2L].[LanguageID]
        LEFT OUTER JOIN [Property] as [Prop] on [Prop].[PropertyID] =
[F].[PrimaryPropertyID]
        LEFT OUTER JOIN [PersontoPhoneNumber] as [P2Ph] on
[P].[PersonID] = [P2Ph].[PersonID]
        LEFT OUTER JOIN [PhoneNumber] as [Ph] on [Ph].[PhoneNumberID] =
[P2Ph].[PhoneNumberID]
        LEFT OUTER JOIN [PhoneNumberType] as [PhT] on [Ph].[Phone-
NumberTypeID] = [PhT].[PhoneNumberTypeID]
        LEFT OUTER JOIN [PersontoHobby] as [P2H] on [P].[PersonID] =
[P2H].[HobbyID]
        LEFT OUTER JOIN [Hobby] as [H] on [H].[HobbyID] =
[P2H].[HobbyID]
        LEFT OUTER JOIN [Lab] on [BTR].[LabID] = [Lab].[LabID]
        WHERE 1 = 1'

if @Last_Name IS NOT NULL
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' AND [p].[LastName] = @LastName
ORDER BY [P].[PersonID] desc'
ELSE
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' ORDER BY [P].[PersonID] desc'

IF @Last_name is NULL
    SET @Recompile = 0;

IF @Recompile = 1
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    EXEC [sp_executesql] @spexecutesqlStr
    , N'@Lastname varchar(50)'
    , @LastName = @Last_name;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

```

```
EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]



[dbo].[usp_SLInsertedDataSimplified]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20130509
-- Description:

-- =====

CREATE PROCEDURE [dbo].[usp_SLInsertedDataSimplified]
    -- Add the parameters for the stored procedure here

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @spexecutesqlStr      NVARCHAR(4000),
            @Recompile BIT = 1, @ErrorLogID int
            , @DEBUG BIT = 0;

    -- Insert statements for procedure here

    BEGIN TRY
        SELECT [P].[PersonID]
        , 'P2FPersonID' = [P2F].[PersonID]
        , 'FamilyLastName' = [F].[Lastname]
        , [F].[FamilyID]
        , 'P2FFamilyID' = [P2F].[FamilyID]
        , [P].[LastName]
        , [P].[MiddleName]
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SLIInsertedData-Simplified

```
, [P].[FirstName]
, [P].[BirthDate]
, [P].[Gender]
--,'StreetAddress' = cast([Prop].[StreetNumber] as varchar)

--      + ' '+ cast([Prop].[Street] as varchar) + ' '

--      + cast([Prop].[StreetSuffix] as varchar)

--, [Prop].[ApartmentNumber]

--, [Prop].[City]

--, [Prop].[State]

--, [Prop].[Zipcode]

--, 'PrimaryPhoneNumber' = [Ph].[PhoneNumber]

--, [L].[LanguageName]

, [F].[NumberofSmokers]
, [F].[Pets]
, [F].[Petsoninout]
, [FN].[Notes]

FROM [Person] AS [P]
FULL OUTER JOIN [PersonstoFamily] as [P2F] on [P].[PersonID] = [P2F].[PersonID]
FULL OUTER JOIN [Family] AS [F] on [F].[FamilyID] = [P2F].[FamilyID]
FULL OUTER JOIN [FamilyNotes] AS [FN] on [F].[FamilyID] = [FN].[FamilyID]
--    FULL OUTER JOIN [PersonstoProperty] as [P2P] on [P].PersonID = [P2P].[PersonID]

--    FULL OUTER JOIN [Property] as [Prop] on [Prop].[PropertyID] = [F].[Primary-
PropertyID]

-- where [P2F].FamilyID is NULL

-- People to families: 3470

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before

    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
        BEGIN
```

Author: liam

```
ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Family]
[dbo].[FamilyNotes]
[dbo].[Person]
[dbo].[PersonstoFamily]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SlLabName]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150304
-- Description: Lists lab names
-- =====

CREATE PROCEDURE [dbo].[usp_SlLabName]
    -- Add the parameters for the stored procedure here

    @DEBUG BIT = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    -- Insert statements for procedure here

    select LabName from Lab where LabName in ('LeadCare II','Tamarac','Quest
Diagnostic','Other')

END
GO
```

Uses

[dbo].[Lab]

[dbo].[usp_SLLlistAllFamilyMembers]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@FamilyID	int	4

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150103
-- Description: stored procedure to list family members
-- =====

CREATE PROCEDURE [dbo].[usp_SLLlistAllFamilyMembers]
    -- Add the parameters for the stored procedure here

    @FamilyID int = NULL
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @spexecutesQLStr NVARCHAR(4000)
        , @Recompile BIT = 1, @ErrorLogID int;

    --IF (@FamilyID IS NULL)

    --BEGIN
    --    RAISERROR ('You must supply at least one parameter.', 11, -1);

```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SLLListAllFamily-Members

```
-- RETURN;

--END;

SELECT @spexecutesQLStr =
N'SELECT [f].[familyid], FamilyName = [f].[lastname],[P].[Last-
Name],[P].[Firstname]  from [person] as [p]
join [persontoFamily]  [p2f] on [p].[personid] = [p2f].[personid]
join [family] AS [f] on [f].[familyid] = [p2f].[familyid]
where 1=1';

IF (@FamilyID IS NOT NULL)
SELECT @spexecutesQLStr = @spexecutesQLStr
+ N' AND [f].[familyID] = @Family_ID';

SELECT @spexecutesQLStr = @spexecutesQLStr
+ N' order by [f].[FamilyID],[f].[lastname]';

IF (@FamilyID IS NULL)
SET @Recompile = 0;

IF @Recompile = 1
SELECT @spexecutesQLStr = @spexecutesQLStr + N' OPTION(RECOMPILE)';

BEGIN TRY
EXEC [sp_executesql] @spexecutesQLStr
, N'@Family_ID int'
, @Family_ID = @FamilyID;
END TRY
BEGIN CATCH
-- Call procedure to print error information.

EXECUTE dbo.uspPrintError;

-- Roll back any active or uncommittable transactions before
-- inserting information in the ErrorLog.

IF XACT_STATE() <> 0
BEGIN
ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIListClientsByCreatedate]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150120
-- Description: User defined stored procedure to
--               select People by created date range
-- =====
CREATE PROCEDURE [dbo].[usp_SIListClientsByCreatedate]
    -- Add the parameters for the stored procedure here

    @StartDate date = NULL,
    @EndDate date = NULL,
    @DEBUG bit = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SILListClientsBy-
Createdate

```
DECLARE @spexecutesqlStr NVARCHAR (4000),
        @Recompile BIT = 1, @ErrorLogID int, @ReturnError int;

--SELECT [P].[PersonID], 'FamilyName' = [F].[LastName]

--     , [P].[LastName], [P].[FirstName], [P].[CreatedDate]

--     FROM [Person] AS [P]

--     JOIN PersonstoFamily AS P2F ON [P].[PersonID] = [P2F].[PersonID]

--     JOIN [family] AS [F] ON [P2F].[FamilyID] = [F].[FamilyID]

--     where 1 = 2 AND [P].[CreatedDate] >= @StartDate AND [P].[CreatedDate] <= @EndDate
--     order by [P].[LastName],[P].[PersonID] OPTION(RECOMPILE)

select @spexecutesqlStr ='SELECT [P].[PersonID], [P].[LastName], [P].[Middle-
Name], [P].[FirstName], [P].[BirthDate]
                           , [P].[Gender], [P].[Age], [P].[ModifiedDate], [P].[Created-
Date]
                           from [Person] AS [P]
                           where 1 = 1'

-- Return all People if nothing was passed in

IF ((@StartDate is NULL) AND (@EndDate is NULL))
    SET @Recompile = 0

IF (@StartDate is NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND [P].[CreatedDate] >= @Begin-
Date'

IF (@EndDate is NOT NULL)
BEGIN
    SET @EndDate = DateAdd(dd,1,@EndDate)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND [P].[CreatedDate] < @End-
Date'
END

-- order by last name

SELECT @spexecutesqlStr = @spexecutesqlStr + N' order by [P].[CreatedDate] DESC,
[P].[LastName],
[P].[PersonID] ASC'

IF @Recompile = 1
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesqlStr, 'BEGINDate' = @StartDate, 'ENDDate' = @EndDate,
'DEBUG' = @Debug

        EXEC [sp_executesql] @spexecutesqlStr
```

Author: liam

```
, N'@BeginDate datetime, @EndDate datetime'
, @BeginDate = @StartDate
, @EndDate = @EndDate;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Add error information to errorlog

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    SELECT @ReturnError = ERROR_NUMBER();

    DROP TABLE ##ReturnedValues;
    RETURN @ReturnError
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIListClientsByModifieddate]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150120
-- Description: User defined stored procedure to
--               select People by created date range
-- =====

CREATE PROCEDURE [dbo].[usp_SIListClientsByModifieddate]
    -- Add the parameters for the stored procedure here

    @StartDate date = NULL,
    @EndDate date = NULL,
    @DEBUG bit = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SIListClientsBy-Modifieddate

```
DECLARE @spexecutesqlStr NVARCHAR (4000),
        @Recompile BIT = 1, @ErrorLogID int, @ReturnError int;

--SELECT [P].[PersonID], 'FamilyName' = [F].[LastName]

--     , [P].[LastName], [P].[FirstName], [P].[CreatedDate]

--     FROM [Person] AS [P]

--     JOIN PersontoFamily AS P2F ON [P].[PersonID] = [P2F].[PersonID]

--     JOIN [family] AS [F] ON [P2F].[FamilyID] = [F].[FamilyID]

--     where 1 = 2 AND [P].[CreatedDate] >= @StartDate AND [P].[CreatedDate] <= @EndDate
--     order by [P].[LastName],[P].[PersonID] OPTION(RECOMPILE)

select @spexecutesqlStr ='SELECT [P].[PersonID],[P].[LastName],[P].[Middle-
Name],[P].[FirstName],[P].[BirthDate],[P].[Gender],[P].[ModifiedDate]
from [Person] AS [P]
where 1 = 1'

-- Return all People if nothing was passed in

IF ((@StartDate is NULL) AND (@EndDate is NULL))
    SET @Recompile = 0

IF (@StartDate is NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND [P].[ModifiedDate] >= @Begin-
Date'

IF (@EndDate is NOT NULL)
    BEGIN
        SET @EndDate = DateAdd(dd,1,@EndDate)
        SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND [P].[ModifiedDate] < @End-
Date'
    END

-- order by last name

SELECT @spexecutesqlStr = @spexecutesqlStr + N' order by [P].[ModifiedDate] ASC,
[P].[LastName],
[P].[PersonID] ASC'

IF @Recompile = 1
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesqlStr, 'BEGINDate' = @StartDate, 'ENDDate' = @EndDate,
'DEBUG' = @Debug

        EXEC [sp_executesql] @spexecutesqlStr
        , N'@BeginDate datetime, @EndDate datetime'
    
```

Author: liam

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SIListClientsBy-Modifieddate

```
, @BeginDate = @StartDate  
, @EndDate = @EndDate;  
END TRY  
BEGIN CATCH  
    -- Call procedure to print error information.  
  
    EXECUTE dbo.uspPrintError;  
  
    -- Add error information to errorlog  
  
    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;  
    SELECT @ReturnError = ERROR_NUMBER();  
  
    DROP TABLE ##ReturnedValues;  
    RETURN @ReturnError  
END CATCH;  
END  
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIListFamilies]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

SQL Script

```
-- =====

-- Author:      Liam Thier

-- Create date: 20150110

-- Description: User defined stored procedure to

--               select all families

-- =====

CREATE PROCEDURE [dbo].[usp_SIListFamilies]
    -- Add the parameters for the stored procedure here

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from

    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @spexecutesqlStr      NVARCHAR (4000),
            @Recompile   BIT = 1, @ErrorLogID int;

    -- Insert statements for procedure here

    select @spexecutesqlStr ='SELECT [F].[FamilyID], ''FamilyName'' = [F].[LastName]
from [family] AS [F]
where 1 = 1'

    -- Return all families and associated properties if nothing was passed in

    SET @Recompile = 0

```

Author: liam

```
-- order by last name

    SELECT @spexecutesqlStr = @spexecutesqlStr + N' order by [F].[LastName], [F].[Family-
ID]'

    IF @Recompile = 1
        SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

    BEGIN TRY
        EXEC [sp_executesql] @spexecutesqlStr;
    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Add error information to errorlog

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIListFamilyMembers]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@FamilyID	int	4

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150103
-- Description: stored procedure to list family members
-- =====

CREATE PROCEDURE [dbo].[usp_SIListFamilyMembers]
    -- Add the parameters for the stored procedure here

    @FamilyID int = NULL
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @spexecutesQLStr NVARCHAR(4000)
        , @Recompile BIT = 1, @ErrorLogID int;

    --IF (@FamilyID IS NULL)

    --BEGIN
    --    RAISERROR ('You must supply at least one parameter.', 11, -1);
    --END
    -- RETURN;
```

Author: liam

```
--END;

SELECT @spexecutesQLStr =
N'SELECT [f].[familyid], FamilyName = [f].[lastname],[P].[Last-
Name],[P].[Firstname]  from [person] as [p]
join [persontoFamily] [p2f] on [p].[personid] = [p2f].[personid]
join [family] AS [f] on [f].[familyid] = [p2f].[familyid]
where 1=1';

IF (@FamilyID IS NOT NULL)
SELECT @spexecutesQLStr = @spexecutesQLStr
+ N' AND [f].[familyID] = @Family_ID';

SELECT @spexecutesQLStr = @spexecutesQLStr
+ N' order by [f].[lastname],[f].[familyid]';

IF (@FamilyID IS NULL)
SET @Recompile = 0;

IF @Recompile = 1
SELECT @spexecutesQLStr = @spexecutesQLStr + N' OPTION(RECOMPILE)';

BEGIN TRY
EXEC [sp_executesql] @spexecutesQLStr
, N'@Family_ID int'
, @Family_ID = @FamilyID;
END TRY
BEGIN CATCH
-- Call procedure to print error information.

EXECUTE dbo.uspPrintError;

-- Roll back any active or uncommittable transactions before
-- inserting information in the ErrorLog.

IF XACT_STATE() <> 0
BEGIN
ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIListNursingWomenbyCreateDateRange]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Begin_Date	date	3
@End_Date	date	3
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150120
-- Description: User defined stored procedure to
--               select NursingWomen by created date range
-- =====

CREATE PROCEDURE [dbo].[usp_SIListNursingWomenbyCreateDateRange]
    -- Add the parameters for the stored procedure here

    @Begin_Date date = NULL,
    @End_Date date = NULL,
    @DEBUG bit = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
```

```
DECLARE @spexecutesqlStr      NVARCHAR (4000),
        @Recompile   BIT = 1, @ErrorLogID int, @ReturnError int;

select @spexecutesqlStr ='SELECT [P].[PersonID], [P].[LastName], [P].[FirstName], [P].[CreatedDate]
                           from [Person] AS [P]
                           where NursingMother = 1'

-- Return all NursingWomen if nothing was passed in

IF ((@Begin_Date is NULL) AND (@End_Date is NULL))
    SET @Recompile = 0

IF (@Begin_Date is NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND [P].[CreatedDate] >= @Begin-
Date'

IF (@End_Date is NOT NULL)
BEGIN
    SET @End_Date = DateAdd(dd,1,@End_Date)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND [P].[CreatedDate] < @End-
Date'
END

-- order by last name

SELECT @spexecutesqlStr = @spexecutesqlStr + N' order by [P].[CreatedDate] DESC,
[P].[LastName],
[P].[PersonID] ASC'

IF @Recompile = 1
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesqlStr, 'BEGINDATE' = @Begin_Date, 'ENDDATE' = @End_Date,
'DEBUG' = @Debug

        EXEC [sp_executesql] @spexecutesqlStr
        , N'@BeginDate datetime, @EndDate datetime'
        , @BeginDate = @Begin_Date
        , @EndDate = @End_Date;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Add error information to errorlog

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    SELECT @ReturnError = ERROR_NUMBER();
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SIListNursing-WomenbyCreateDateRange

```
DROP TABLE ##ReturnedValues;
RETURN @ReturnError
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIListPeoplebyCreateDateRange]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Begin_Date	date	3
@End_Date	date	3
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150120
-- Description: User defined stored procedure to
--               select People by created date range
-- =====
CREATE PROCEDURE [dbo].[usp_SIListPeoplebyCreateDateRange]
    -- Add the parameters for the stored procedure here
    @Begin_Date date = NULL,
    @End_Date date = NULL,
    @DEBUG bit = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SILListPeoplebyCreateDateRange

```
DECLARE @spexecutesqlStr NVARCHAR (4000),
        @Recompile BIT = 1, @ErrorLogID int, @ReturnError int;

--SELECT [P].[PersonID], 'FamilyName' = [F].[LastName]

--     , [P].[LastName], [P].[FirstName], [P].[CreatedDate]

--     FROM [Person] AS [P]

--     JOIN PersonstoFamily AS P2F ON [P].[PersonID] = [P2F].[PersonID]

--     JOIN [family] AS [F] ON [P2F].[FamilyID] = [F].[FamilyID]

--     where 1 = 2 AND [P].[CreatedDate] >= @Begin_Date AND [P].[CreatedDate] <=
@End_Date order by [P].[LastName], [P].[PersonID] OPTION(RECOMPILE)

select @spexecutesqlStr ='SELECT [P].[PersonID], [P].[LastName], [P].[First-
Name], [P].[CreatedDate]
from [Person] AS [P]
where 1 = 1'

-- Return all People if nothing was passed in

IF ((@Begin_Date is NULL) AND (@End_Date is NULL))
    SET @Recompile = 0

IF (@Begin_Date is NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND [P].[CreatedDate] >= @Begin-
Date'

IF (@End_Date is NOT NULL)
    BEGIN
        SET @End_Date = DateAdd(dd,1,@End_Date)
        SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND [P].[CreatedDate] < @End-
Date'
    END

-- order by last name

SELECT @spexecutesqlStr = @spexecutesqlStr + N' order by [P].[CreatedDate] DESC,
[P].[LastName],
[P].[PersonID] ASC'

IF @Recompile = 1
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesqlStr, 'BEGINDate' = @Begin_Date, 'EndDate' = @End_Date,
'DEBUG' = @Debug

        EXEC [sp_executesql] @spexecutesqlStr
        , N'@BeginDate datetime, @EndDate datetime'
    
```

Author: liam

```
, @BeginDate = @Begin_Date
, @EndDate = @End_Date;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Add error information to errorlog

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    SELECT @ReturnError = ERROR_NUMBER();

    DROP TABLE ##ReturnedValues;
    RETURN @ReturnError
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SLLListPotentialDuplicatePeople]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Debug	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150127
-- Description: stored procedure to potential
--               duplicate people
-- =====
CREATE PROCEDURE [dbo].[usp_SLLListPotentialDuplicatePeople]
    -- Add the parameters for the stored procedure here
    @Debug bit = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @spexecutesQLStr NVARCHAR(4000)
        , @Recompile BIT = 1, @ErrorLogID int;

    SELECT @spexecutesQLStr =
        N'SELECT P1PersonID = P1.PersonID
            , P2PersonID = P2.PersonID
```

Author: liam

```
, P1LastName = P1.LastName
, P2LastName = P2.LastName
, P1FirstName = P1.FirstName
, P2FirstName = P2.FirstName
, P1BirthDate = P1.BirthDate
, P2BirthDate = P2.BirthDate
, P1Gender = P1.Gender
, P2Gender = P2.Gender
, P1CreatedDate = P1.CreatedDate
, P2CreatedDate = P2.CreatedDate
, P1ModifiedDate = P1.ModifiedDate
, P2ModifiedDate = P2.ModifiedDate
from person AS P1
JOIN person AS P2 on
    P1.LastName = P2.LastName
    AND P1.FirstName = P2.FirstName
    AND P1.Age = P2.Age
    AND P1.PersonID != P2.PersonID
    OPTION (RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesql;
    EXEC [sp_executesql] @spexecutesql;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Log Errors

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SLLListPotentialDuplicateProperties]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Debug	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150127
-- Description: stored procedure to potential
--               duplicate properties
-- =====

CREATE PROCEDURE [dbo].[usp_SLLListPotentialDuplicateProperties]
    -- Add the parameters for the stored procedure here

    @Debug bit = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @spexecutesQLStr NVARCHAR(4000)
        , @Recompile BIT = 1, @ErrorLogID int;

    SELECT @spexecutesQLStr =
        N'SELECT [P1PropertyID] = [P1].[PropertyID]
            , [P2PropertyID] = [P2].[PropertyID]
```

Author: liam

```
, [P1StreetNumber] = [P1].[StreetNumber]
, [P2StreetNumber] = [P2].[StreetNumber]
, [P1Street] = [P1].[Street]
, [P2Street] = [P2].[Street]
, [P1StreetSuffix] = [P1].[StreetSuffix]
, [P2StreetSuffix] = [P2].[StreetSuffix]
, [P1City] = [P1].[City]
, [P2City] = [P2].[City]
, [P1State] = [P1].[State]
, [P2State] = [P2].[State]
, [P1ZipCode] = [P1].[Zipcode]
, [P2ZipCode] = [P2].[Zipcode]
, [P1County] = [P1].[County]
, [P2County] = [P2].[County]
, [P1CreatedDate] = [P1].[CreatedDate]
, [P2CreatedDate] = [P2].[CreatedDate]
, [P1ModifiedDate] = [P1].[ModifiedDate]
, [P2ModifiedDate] = [P2].[ModifiedDate]
from [Property] AS [P1]
JOIN [Property] AS [P2] on
    [P1].[Street] = [P2].[Street]
    AND [P1].[StreetNumber] = [P2].[StreetNumber]
    AND [P1].[City] = [P2].[City]
    AND [P1].[County] = [P2].[County]
    AND [P1].[Zipcode] = [P2].[Zipcode]
    AND [P1].[State] = [P2].[State]
    AND [P1].[PropertyID] != [P2].[PropertyID]
OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesQLStr;
    EXEC [sp_executesql] @spexecutesQLStr;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Log Errors

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

Author: liam

[dbo].[usp_SIListPregnantWomenbyCreateDateRange]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Begin_Date	date	3
@End_Date	date	3
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150120
-- Description: User defined stored procedure to
--               select PregnantWomen by created date range
-- =====

CREATE PROCEDURE [dbo].[usp_SIListPregnantWomenbyCreateDateRange]
    -- Add the parameters for the stored procedure here

    @Begin_Date date = NULL,
    @End_Date date = NULL,
    @DEBUG bit = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
```

Author: liam

```
DECLARE @spexecutesqlStr      NVARCHAR (4000),
        @Recompile   BIT = 1, @ErrorLogID int, @ReturnError int;

select @spexecutesqlStr ='SELECT [P].[PersonID], [P].[LastName], [P].[FirstName], [P].[CreatedDate]
                           from [Person] AS [P]
                           where Pregnant = 1

-- Return all PregnantWomen if nothing was passed in

IF ((@Begin_Date is NULL) AND (@End_Date is NULL))
    SET @Recompile = 0

IF (@Begin_Date is NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND [P].[CreatedDate] >= @Begin-
Date'

IF (@End_Date is NOT NULL)
BEGIN
    SET @End_Date = DateAdd(dd,1,@End_Date)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND [P].[CreatedDate] < @End-
Date'
END

-- order by last name

SELECT @spexecutesqlStr = @spexecutesqlStr + N' order by [P].[CreatedDate] DESC,
[P].[LastName],
[P].[PersonID] ASC'

IF @Recompile = 1
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesqlStr, 'BEGINDATE' = @Begin_Date, 'ENDDATE' = @End_Date,
'DEBUG' = @Debug

        EXEC [sp_executesql] @spexecutesqlStr
        , N'@BeginDate datetime, @EndDate datetime'
        , @BeginDate = @Begin_Date
        , @EndDate = @End_Date;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Add error information to errorlog

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    SELECT @ReturnError = ERROR_NUMBER();
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SIListPregnant-WomenbyCreateDateRange

```
--      DROP TABLE ##ReturnedValues;

      RETURN @ReturnError
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SLMostRecentBloodTestResults]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Person_ID	int	4
@Min_Lead_Value	numeric(4,1)	5
@Max_Lead_Value	numeric(4,1)	5
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20141222
-- Description: select most recent blood test results
--               optionally only return for a specific
--               client
-- =====
CREATE PROCEDURE [dbo].[usp_SLMostRecentBloodTestResults]
    -- Add the parameters for the stored procedure here
    @Person_ID int = NULL,
    @Min_Lead_Value numeric(4,1) = NULL,
    @Max_Lead_Value numeric(4,1) = NULL,
    @DEBUG bit = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @spexecutesqlStr      NVARCHAR(4000), @OrderBy NVARCHAR(500),
        @Recompile BIT = 1, @ErrorLogID int;

-- Insert statements for procedure here

SELECT @spexecutesqlStr = N'Select [P].[LastName], [P].[FirstName], [P].[Person-
ID], [BTR].[LeadValue], [BTR].[SampleDate], [BTR].[HemoglobinValue]
          , [BTR].[CreatedDate], [BTR].[ModifiedDate], [BTR].[Blood-
TestResultsID] from [Person] AS [P]
          JOIN [BloodTestResults] AS [BTR] on [BTR].[BloodTest-
ResultsID] = (
          select top 1 [BloodTestResultsID] from [BloodTest-
Results]
          where [BloodTestResults].[PersonID] = [P].[Person-
ID]
          -- AND [LeadValue] > @MinLeadValue uncomment to
list most recent tests with BLL above minimum
          )
WHERE 1=1';

IF @Min_Lead_Value IS NULL
    SET @Min_Lead_Value = 0.0;

IF @Person_ID IS NOT NULL
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' AND [p].[PersonID] = @Person-
ID';

IF (@Min_Lead_Value > 0)
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' AND [BTR].[LeadValue] >= @Min-
LeadValue';

IF (@Max_Lead_Value > 0)
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' AND [BTR].[LeadValue] < @Max-
LeadValue';

IF @Person_ID is NULL
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' ORDER BY [p].[LastName],
[P].[PersonID] ASC, [BTR].[SampleDate] DESC';

IF ( (@Person_ID IS NULL) AND (@Min_Lead_Value = 0) )
    SET @Recompile = 0;

IF @Recompile = 1
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    -- If debugging print out query

    IF (@DEBUG = 1)
        SELECT @spexecutesqlStr, 'PID' = @Person_ID, 'MLV' = @Min_Lead_Value, 'Max-
LV' = @Max_Lead_Value, 'R' = @Recompile;
```

```
EXEC [sp_executesql] @spexecutesqlStr
, N'@PersonID int,@MinLeadValue numeric(4,1), @MaxLeadvalue numeric(4,1)'
, @PersonID = @Person_ID, @MinLeadValue = @Min_Lead_Value, @MaxleadValue =
@Max_Lead_Value;
END TRY
BEGIN CATCH
-- Call procedure to print error information.

EXECUTE dbo.uspPrintError;

-- Roll back any active or uncommittable transactions before
-- inserting information in the ErrorLog.

IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIPersonNotes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@DEBUG	bit	1

SQL Script

```
-- =====

-- Author:      Liam Thier

-- Create date: 20150215

-- Description: stored procedure to list

--          person and their ethnicities

-- =====

CREATE PROCEDURE [dbo].[usp_SIPersonNotes]
    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @DEBUG BIT = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @spexecutesQLStr NVARCHAR(4000)
        , @Recompile BIT = 1, @ErrorLogID int;

    --IF (@FamilyID IS NULL)
```

Author: liam

```
--BEGIN

--      RAISERROR ('You must supply at least one parameter.', 11, -1);

--      RETURN;

--END;

SELECT @spexecutesQLStr =
N'select P.PersonID,LastName,FirstName, PN.Notes,P.ModifiedDate from Person AS
P
      LEFT OUTER JOIN PersonNotes AS PN on P.PersonID = PN.PPersonID
      where Notes is not null';

IF (@PersonID IS NOT NULL)
    SELECT @spexecutesQLStr = @spexecutesQLStr
        + N' AND [P].[PersonID] = @PersonID';

SELECT @spexecutesQLStr = @spexecutesQLStr
    + N' order by [P].[lastname],[P].[Personid]';

IF (@PersonID IS NULL)
    SET @Recompile = 0;

IF @Recompile = 1
    SELECT @spexecutesQLStr = @spexecutesQLStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesQLStr, 'PersonID' = @PersonID;

    EXEC [sp_executesql] @spexecutesQLStr
        , N'@PersonID int'
        , @PersonID = @PersonID;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
```

```
END  
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIPersontoEthnicity]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150215
-- Description: stored procedure to list
--               person and their ethnicities
-- =====

CREATE PROCEDURE [dbo].[usp_SIPersontoEthnicity]
    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @DEBUG BIT = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @spexecutesQLStr NVARCHAR(4000)
           , @Recompile BIT = 1, @ErrorLogID int;

    --IF (@FamilyID IS NULL)
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SIPersontoEthnicity

```
--BEGIN

--      RAISERROR ('You must supply at least one parameter.', 11, -1);

--      RETURN;

--END;

SELECT @spexecutesQLStr =
N'select P.PersonID,LastName,FirstName,E.Ethnicity from Person AS P
    LEFT OUTER JOIN PersontoEthnicity AS P2E on P.PersonID = P2E.PErsonID
    LEFT OUTER JOIN Ethnicity AS E on P2E.EthnicityID = E.EthnicityID
    WHERE 1 = 1';

IF (@PersonID IS NOT NULL)
    SELECT @spexecutesQLStr = @spexecutesQLStr
        + N' AND [P].[PersonID] = @PersonID';

SELECT @spexecutesQLStr = @spexecutesQLStr
    + N' order by [P].[lastname],[P].[Personid]';

IF (@PersonID IS NULL)
    SET @Recompile = 0;

IF @Recompile = 1
    SELECT @spexecutesQLStr = @spexecutesQLStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesQLStr, 'PersonID' = @PersonID;

    EXEC [sp_executesql] @spexecutesQLStr
        , N'@PersonID int'
        , @PersonID = @PersonID;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()

```

Author: liam

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SIPersonto-Ethnicity

```
    END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIPersontoLanguage]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150215
-- Description: stored procedure to list
--               person and their languages
-- =====

CREATE PROCEDURE [dbo].[usp_SIPersontoLanguage]
    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @DEBUG BIT = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
    DECLARE @spexecutesQLStr NVARCHAR(4000)
           , @Recompile BIT = 1, @ErrorLogID int;

    --IF (@FamilyID IS NULL)
```

```
--BEGIN

--      RAISERROR ('You must supply at least one parameter.', 11, -1);

--      RETURN;

--END;

SELECT @spexecutesQLStr =
N'select [P].PersonID,LastName,FirstName, L.LanguageName from Person AS P
    LEFT OUTER JOIN PersonToLanguage AS P2L on P.PersonID = P2L.PErsoneID
    LEFT OUTER JOIN Language AS L on P2L.LanguageID = L.LanguageID
    WHERE 1 = 1';

IF (@PersonID IS NOT NULL)
    SELECT @spexecutesQLStr = @spexecutesQLStr
        + N' AND [P].[PersonID] = @PersonID';

SELECT @spexecutesQLStr = @spexecutesQLStr
    + N' order by [P].[lastname],[P].[Personid]';

IF (@PersonID IS NULL)
    SET @Recompile = 0;

IF @Recompile = 1
    SELECT @spexecutesQLStr = @spexecutesQLStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    IF (@DEBUG = 1)
        SELECT @spexecutesQLStr, 'PersonID' = @PersonID;

    EXEC [sp_executesql] @spexecutesQLStr
        , N'@PersonID int'
        , @PersonID = @PersonID;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SIPersonto-Language

```
    END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIRelationshipTypes]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150110
-- Description: User defined stored procedure to
--               select all relationship types and IDs
-- =====
CREATE PROCEDURE [dbo].[usp_SIRelationshipTypes]
    -- Add the parameters for the stored procedure here
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @spexecutesqlStr NVARCHAR (4000),
            @Recompile BIT = 1, @ErrorLogID int;

    -- Insert statements for procedure here

    select @spexecutesqlStr ='SELECT [RT].[RelationshipTypeID], [RT].[RelationshipTypeName]
        from [RelationshipType] AS [RT]
        where 1 = 1'

    -- Return all families and associated properties if nothing was passed in
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SIRelationshipTypes

```
SET @Recompile = 0

-- order by last name

SELECT @spexecutesqlStr = @spexecutesqlStr + N' order by [RT].[RelationshipType-
Name]'

IF @Recompile = 1
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

BEGIN TRY
    EXEC [sp_executesql] @spexecutesqlStr;
END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Add error information to errorlog

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER();
END CATCH;
END
GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SIStatus]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@TargetType	varchar(50)	50
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150215
-- Description:    returns valid status codes for passed in type - Child
-- =====

CREATE PROCEDURE [dbo].[usp_SIStatus]
    -- Add the parameters for the stored procedure here

    @TargetType varchar(50) = NULL,
    @DEBUG     BIT    = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    -- Insert statements for procedure here

    -- IF (@StatusType = 'Child')

        select statusName from TargetStatus where TargetType = @TargetType
END
```

Author: liam

```
GO
```

Uses

[dbo].[TargetStatus]

[dbo].[usp_SISummaryReport]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@MinLeadValue	numeric(4,1)	5
@MaxLeadValue	numeric(4,1)	5
@MinAge	int	4
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150605
-- Description: procedure returns the number of
--               blood tests conducted within
--               the specified date range.
-- =====

CREATE PROCEDURE [dbo].[usp_SISummaryReport]
    -- Add the parameters for the stored procedure here

    @StartDate date = NULL,
    @EndDate date = NULL,
    @MinLeadValue numeric(4,1) = NULL,
    @MaxLeadValue Numeric(4,1) = NULL,
    @MinAge int = 18,
```

```

@DEBUG bit = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @spexecutesqlStr NVARCHAR(4000), @Recompile BIT = 1, @ErrorLogID int, @Parm-
Definition nvarchar(500)
        , @ClientCount int, @NewClientCount int, @BLLCount int, @EBLLCount int, @Pregnant-
WomenCount int
        , @NursingMotherCount int, @NursingInfantCount int, @AdultCount int, @BloodTest-
Count int, @HomeSoilCount int;

    BEGIN TRY

        IF (@StartDate IS NULL)
            SET @StartDate = '18000101';

        IF (@EndDate IS NULL)
            SET @EndDate = GETDATE();

        IF (@StartDate >= @EndDate)
            BEGIN
                DECLARE @ErrorMessage VARCHAR(3000);
                SET @ErrorMessage ='EndDate must be after StartDate: StartDate: ' +
cast(@StartDate as varchar) + ' EndDate: ' + cast(@EndDate as varchar)
                RAISERROR (@ErrorMessage, 11, -1);
                RETURN;
            END

        -- clients

        SELECT @spexecutesqlStr = 'Select @Clients = count(PersonID) from (
            SELECT PersonID from bloodTestResults WHERE 1=1 AND SampleDate >= @Start-
Date AND SampleDate < @EndDate
            UNION
            SELECT PersonID from Questionnaire WHERE 1=1 AND QuestionnaireDate >=
@StartDate AND QuestionnaireDate < @EndDate
        ) total'

        IF (@Recompile = 1)
            SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

        IF (@DEBUG = 1)
            SELECT @spexecutesqlStr, [StartDate] = @StartDate, [EndDate] = @EndDate

        EXEC [sp_executesql] @spexecutesqlStr
        , N'@StartDate date, @EndDate date, @Clients int OUTPUT'
        , @StartDate = @StartDate
        , @EndDate = @EndDate
        , @Clients = @ClientCount OUTPUT;
    
```

```

-- NewClients

SELECT @spexecutesqlStr = 'Select @NewClients = count(PersonID) from Person
WHERE isClient = 1'

IF (@StartDate IS NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND CreatedDate >= @Start-
Date'

IF (@EndDate IS NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND CreatedDate < @EndDate'

IF (@Recompile = 1)
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

IF (@DEBUG = 1)
    SELECT @spexecutesqlStr, [StartDate] = @StartDate, [EndDate] = @EndDate

EXEC [sp_executesql] @spexecutesqlStr
, N'@StartDate date, @EndDate date, @NewClients int OUTPUT'
, @StartDate = @StartDate
, @EndDate = @EndDate
, @NewClients = @NewClientCount OUTPUT

-- Total BloodLead Tests

SELECT @spexecutesqlStr = 'SELECT @BloodTestCount = count([BloodTestResultsID])
from [BloodTestResults]
where 1 = 1'

IF (@StartDate IS NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND SampleDate >= @Start-
Date'

IF (@EndDate IS NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND SampleDate < @EndDate'

IF (@Recompile = 1)
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

IF (@DEBUG = 1)
    SELECT @spexecutesqlStr, [StartDate] = @StartDate, [EndDate] = @EndDate

EXEC [sp_executesql] @spexecutesqlStr
, N'@StartDate date, @EndDate date, @BloodTestCount int OUTPUT'
, @StartDate = @StartDate
, @EndDate = @EndDate
, @BloodTestCount = @BloodTestCount OUTPUT

-- BLL 5 ug/dl - 9.9 ug/dl

SET @MinLeadValue = 5.0;
SET @MaxLeadValue = 10.0;

```

```

SELECT @spexecutesqlStr = 'SELECT @BLLCount = count([BloodTestResultsID]) from
[BloodTestResults]
      where 1 = 1'

IF (@MinLeadValue IS NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND Leadvalue >= @MinLead-
Value'

IF (@MaxLeadValue IS NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND LeadValue < @MaxLead-
Value'

IF (@StartDate IS NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND SampleDate >= @Start-
Date'

IF (@EndDate IS NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND SampleDate < @EndDate'

IF (@Recompile = 1)
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

IF (@DEBUG = 1)
    SELECT @spexecutesqlStr, [StartDate] = @StartDate, [EndDate] = @EndDate,
[MinLeadValue] = @MinLeadValue, [MaxLeadValue] = @MaxLeadValue

EXEC [sp_executesql] @spexecutesqlStr
    , N'@StartDate date, @EndDate date, @MinLeadValue numeric(4,1), @MaxLeadValue
numeric(4,1), @BLLCount int OUTPUT'
    , @StartDate = @StartDate
    , @EndDate = @EndDate
    , @MinLeadValue = @MinLeadValue
    , @MaxLeadValue = @MaxLeadValue
    , @BLLCount = @BLLCount OUTPUT

-- BLL 10 ug/dl and above

SET @MinLeadValue = 10;
SET @MaxLeadValue = NULL;

SELECT @spexecutesqlStr = 'SELECT @EBLLCount = count([BloodTestResultsID]) from
[BloodTestResults]
      where 1 = 1'

IF (@MinLeadValue IS NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND Leadvalue >= @MinLead-
Value'

IF (@MaxLeadValue IS NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND LeadValue < @MaxLead-
Value'

IF (@StartDate IS NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND SampleDate >= @Start-
Date'

```

```

IF (@EndDate IS NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND SampleDate < @EndDate'

IF (@Recompile = 1)
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

IF (@DEBUG = 1)
    SELECT @spexecutesqlStr, [StartDate] = @StartDate, [EndDate] = @EndDate,
[MinLeadValue] = @MinLeadValue, [MaxLeadValue] = @MaxLeadValue

EXEC [sp_executesql] @spexecutesqlStr
, N'@StartDate date, @EndDate date, @MinLeadValue numeric(4,1), @MaxLeadValue
numeric(4,1), @EBLLCount int OUTPUT'
, @StartDate = @StartDate
, @EndDate = @EndDate
, @MinLeadValue = @MinLeadValue
, @MaxLeadValue = @MaxLeadValue
, @EBLLCount = @EBLLCount OUTPUT

-- Pregnant women

select @spexecutesqlStr ='Select @PregnantWomen = COUNT(PersonID) from (
                    Select BTR.PersonID,Q.Pregnant from BloodTestResults AS
BTR
                    LEFT OUTER JOIN [Questionnaire] AS [Q] on
[Q].[QuestionnaireID] =
[QuestionnaireID] from [Questionnaire]
                    select top 1
                    where
[Questionnaire].PersonID = [BTR].PersonID
                    AND QuestionnaireDate
>= @StartDate AND QuestionnaireDate < @EndDate
                    order by Pregnant desc
)
                    where SampleDate >= @StartDate and SampleDate < @End-
Date AND Q.Pregnant = 1

UNION
                    SELECT PersonID,Pregnant from Questionnaire where
QuestionnaireDate >= @StartDate and QuestionnaireDate < @EndDate
                    AND Pregnant = 1
                    ) ClientsinReportingPeriod
                    where ClientsinReportingPeriod.Pregnant = 1'

IF @Recompile = 1
    SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

IF (@DEBUG = 1)
    SELECT @spexecutesqlStr, 'StartDate' = @StartDate, 'EndDate' = @EndDate,
'DEBUG' = @Debug

EXEC [sp_executesql] @spexecutesqlStr
, N'@StartDate datetime, @EndDate datetime, @PregnantWomen int OUTPUT'
, @StartDate = @StartDate
, @EndDate = @EndDate

```

```

    , @PregnantWomen = @PregnantWomenCount OUTPUT;

    -- Nursing Mothers

    SELECT @spexecutesqlStr = 'Select @NursingMothers = COUNT(PersonID) from (
                                Select BTR.PersonID,Q.NursingMother from BloodTest-
Results AS BTR
                                LEFT OUTER JOIN [Questionnaire] AS [Q] on
[Q].[QuestionnaireID] =
                                select top 1
[QuestionnaireID] from [Questionnaire]
                                where
[Questionnaire].[PersonID] = [BTR].[PersonID]
                                AND QuestionnaireDate
>= @StartDate AND QuestionnaireDate < @EndDate
                                order by NursingMother
desc
)
where SampleDate >= @StartDate and SampleDate < @End-
Date AND Q.NursingMother = 1

        UNION
        SELECT PersonID,NursingMother from Questionnaire where
QuestionnaireDate >= @StartDate and QuestionnaireDate < @EndDate
        AND NursingMother = 1
    ) ClientsinReportingPeriod
    where ClientsinReportingPeriod.NursingMother = 1'

    IF ((DateDiff(YYYY,@StartDate,@EndDate) > 5))
        SET @Recompile = 0;

    IF (@Recompile = 1)
        SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

    IF (@DEBUG = 1)
        SELECT @spexecutesqlStr, [StartDate] = @StartDate, [EndDate] = @EndDate

        EXEC [sp_executesql] @spexecutesqlStr
        , N'@StartDate date, @EndDate date, @NursingMothers int OUTPUT'
        , @StartDate = @StartDate
        , @EndDate = @EndDate
        , @NursingMothers = @NursingMotherCount OUTPUT

    -- Nursing Infants

    SELECT @spexecutesqlStr = 'Select @NursingInfants = COUNT(PersonID) from (
                                Select BTR.PersonID,Q.NursingInfant from BloodTest-
Results AS BTR
                                LEFT OUTER JOIN [Questionnaire] AS [Q] on
[Q].[QuestionnaireID] =
                                select TOP 1
[QuestionnaireID] from [Questionnaire]
                                where
[Questionnaire].[PersonID] = [BTR].[PersonID]
                                AND Questionnaire-
Date >= @StartDate AND QuestionnaireDate < @EndDate
                                order by Nursing-

```

```

Infant desc
)
where SampleDate >= @StartDate and SampleDate <
@EndDate AND Q.NursingInfant = 1

UNION
SELECT PersonID,NursingInfant from Questionnaire
where QuestionnaireDate >= @StartDate and QuestionnaireDate < @EndDate
AND NursingInfant = 1
) ClientsinReportingPeriod
where ClientsinReportingPeriod.NursingInfant = 1'

IF ((DateDiff(YYYY,@StartDate,@EndDate) > 5))
SET @Recompile = 0;

IF (@Recompile = 1)
SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

IF (@DEBUG = 1)
SELECT @spexecutesqlStr, [StartDate] = @StartDate, [EndDate] = @EndDate

EXEC [sp_executesql] @spexecutesqlStr
, N'@StartDate date, @EndDate date, @NursingInfants int OUTPUT'
, @StartDate = @StartDate
, @EndDate = @EndDate
, @NursingInfants = @NursingInfantCount OUTPUT

-- Adults

IF (@StartDate IS NULL)
SET @StartDate = '18000101'

IF (@EndDate IS NULL)
SET @EndDate = GetDate();

-- Create temporary table

CREATE Table #TempPotentialAdults
( PersonID int
, TestID int
, AgeAtVisit tinyint
, MostRecentVisit date
, Birthdate date
, Visits tinyint
)

-- insert values from bloodtest results

insert Into #TempPotentialAdults (PersonID, MostRecentVisit, TestID)
select PersonID,MostRecentVisit = SampleDate, TestID = BloodTestResults-
ID
from BloodtestResults
where SampleDate >= @StartDate AND SampleDate < @EndDate

```

```

-- insert values from questionnaire

    insert Into #TempPotentialAdults (PersonID, MostRecentVisit, TestID)
        Select PersonID,MostRecentVisit = QuestionnaireDate, TestID =
QuestionnaireID
            from Questionnaire
                where QuestionnaireDate >= @StartDate AND QuestionnaireDate < @End-
Date
                    and (ISNULL(Questionnaire.NursingMother,0) = 0 OR
ISNULL(Questionnaire.Pregnant,0) = 0 )

    -- populate birthdate only if the difference from most recent visit to
birthdate is at least minAge

        update #TempPotentialAdults set BirthDate = Person.Birthdate,
            AgeAtVisit = [dbo].[udf_CalculateAge]([Person].[BirthDate],MostRecent-
Visit)
        FROM #TempPotentialAdults
        JOIN Person on Person.PersonID = #TempPotentialAdults.PersonID
        where Datediff(yy,Person.BirthDate,MostRecentVisit) > @MinAge

    Select @AdultCount = count(distinct PersonID) from #TempPotentialAdults
    where AgeAtVisit > @MinAge

    drop table #TempPotentialAdults

    -- Home visits and soil testing

    select @spexecutesqlStr ='select @HomeSoilCount = count(PersonID) from (
        SELECT PersonID
            from BloodTestResults where SampleDate >= @StartDate and Sample-
Date < @EndDate
                AND ClientStatusID in (    SELECT [TS].[StatusID] from
[TargetStatus] AS [TS]
                                where TargetType = ''Person''
                                AND StatusName in (''Home
visit'', ''Home Visit and Soil Sample'', ''Soil Sample'')
                            )
        UNION
        -- people with questionnaire but no blood test during reporting
period
        Select Q.PersonID
            from Questionnaire AS Q
                LEFT OUTER JOIN [BloodTestResults] AS [BTR] on
[BTR].[BloodTestResultsID] = (
                                select top 1 [BloodTest-
ResultsID] from [BloodTestResults]
                                where [BloodTest-
Results].[PersonID] = [Q].[PersonID]
                                -- AND SampleDate >=
@StartDate AND SampleDate < @EndDate
                                AND BTR.ClientStatusID
                                in (    SELECT
[TS].[StatusID] from [TargetStatus] AS [TS]
                                where
TargetType = ''Person'''
                                AND Status-

```

```

Name in (''Home visit'', ''Home Visit and Soil Sample'', ''Soil Sample'')
)
order by SampleDate
desc
)
where QuestionnaireDate >= @StartDate and Questionnaire-
Date < @EndDate
AND BTR.ClientStatusID
in (    SELECT [TS].[StatusID] from [TargetStatus]
AS [TS]
where TargetType = ''Person''
AND StatusName in (''Home visit'', ''Home
Visit and Soil Sample'', ''Soil Sample'')
)
) HomeVisitSoilSamples'

IF @Recompile = 1
SELECT @spexecutesqlStr = @spexecutesqlStr + N' OPTION(RECOMPILE)';

IF (@DEBUG = 1)
SELECT @spexecutesqlStr, 'StartDate' = @StartDate, 'EndDate' = @EndDate,
'DEBUG' = @Debug

EXEC [sp_executesql] @spexecutesqlStr
, N'@StartDate datetime, @EndDate datetime, @HomeSoilCount int OUTPUT'
, @StartDate = @StartDate
, @EndDate = @EndDate
, @HomeSoilCount = @HomeSoilCount OUTPUT;
-- total tests

select 'ClientCount' = @ClientCount, 'NewClientCount' = @NewClientCount, 'Blood-
TestCount' = @BloodTestCount
, 'BLL5to10ugPerdl' = @BLLCount, 'EBLLCount' = @EBLLCount, 'Pregnant-
Women' = @PregnantWomenCount
, 'NursingMotherCount' = @NursingMotherCount, 'NursingInfantCount' =
@NursingInfantCount
, 'AdultCount' = @AdultCount, 'HomeSoilCount' = @HomeSoilCount;

--      SELECT @sSQL = N'SELECT @retvalOUT = MAX(PersonID) FROM ' + @tablename;

--SET @ParmDefinition = N'@retvalOUT int OUTPUT';

--EXEC sp_executesql @sSQL, @ParmDefinition, @retvalOUT=@retval OUTPUT;

--SELECT @retval;

END TRY
BEGIN CATCH
-- Call procedure to print error information.

EXECUTE dbo.uspPrintError;

```

```
-- Roll back any active or uncommittable transactions before  
  
-- inserting information in the ErrorLog.  
  
IF XACT_STATE() <> 0  
BEGIN  
    ROLLBACK TRANSACTION;  
END  
  
EXECUTE dbo.uspLogError @ErrorLogID = @ErrorLogID OUTPUT;  
RETURN ERROR_NUMBER()  
END CATCH;  
END  
GO
```

Uses

[dbo].[BloodTestResults]
[dbo].[Person]
[dbo].[Questionnaire]
[dbo].[uspLogError]
[dbo].[uspPrintError]
[dbo].[udf_CalculateAge]

[dbo].[usp_SISummaryReport_MetaData]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@StartDate	date	3
@EndDate	date	3
@MinLeadValue	numeric(4,1)	5
@MaxLeadValue	numeric(4,1)	5
@MinAge	int	4
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150605
-- Description: procedure returns the number of
--               blood tests conducted within
--               the specified date range.
-- =====
CREATE PROCEDURE [dbo].[usp_SISummaryReport_MetaData]
    -- Add the parameters for the stored procedure here

    @StartDate date = NULL,
    @EndDate date = NULL,
    @MinLeadValue numeric(4,1) = NULL,
    @MaxLeadValue Numeric(4,1) = NULL,
```

Author: liam

```
@MinAge int = 18,
@DEBUG bit = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @spexecutesqlStr NVARCHAR(4000), @Recompile BIT = 1, @ErrorLogID int, @Parm-
Definition nvarchar(500)
        , @ClientCount int, @NewClientCount int, @BLLCount int, @EBLLCount int, @Pregnant-
WomenCount int
        , @NursingMotherCount int, @NursingInfantCount int, @AdultCount int, @BloodTest-
Count int, @HomeSoilCount int;

    BEGIN TRY

        select 'ClientCount' = @ClientCount, 'NewClientCount' = @NewClientCount, 'Blood-
TestCount' = @BloodTestCount
            , 'BLL5to10ugPerdl' = @BLLCount, 'EBLLCount' = @EBLLCount, 'Pregnant-
Women' = @PregnantWomenCount
            , 'NursingMotherCount' = @NursingMotherCount, 'NursingInfantCount' =
@NursingInfantCount
            , 'AdultCount' = @AdultCount, 'HomeSoilCount' = @HomeSoilCount
            where 1 = 0;

        --      SELECT @sSQL = N'SELECT @retvalOUT = MAX(PersonID) FROM ' + @tablename;
        --SET @ParmDefinition = N'@retvalOUT int OUTPUT';

        --EXEC sp_executesql @sSQL, @ParmDefinition, @retvalOUT=@retval OUTPUT;

        --SELECT @retval;

    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
        BEGIN
```

```
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_SITargetSampleType]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Sample_Target	varchar(50)	50
@p2	int	4

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150102
-- Description: retrieve sample types for people (lead levels)
-- =====
CREATE PROCEDURE [dbo].[usp_SITargetSampleType]
    -- Add the parameters for the stored procedure here
    @Sample_Target varchar(50) = NULL,
    @p2 int = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @spexecutesqlStr nvarchar(4000), @RECOMPILE bit =1;
    -- Insert statements for procedure here

    SELECT @spexecutesqlStr = 'SELECT [SampleTypeID],[SampleTypeName] from [SampleType]
where 1=1'
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_SITargetSample-Type

```
if (@Sample_Target IS NOT NULL)
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' AND [SampleType].[SampleTarget]
= @SampleTarget'

IF @Recompile = 1
    SELECT @spexecutesqlStr = @spexecutesqlStr + ' OPTION(RECOMPILE)';

EXEC [sp_executesql] @spexecutesqlStr
    , N'@SampleTarget varchar(50)', @SampleTarget = @Sample_Target
END
GO
```

[dbo].[usp_upBloodTestResults]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@BloodTestResultsID	int	4
@New_Sample_Date	date	3
@New_Lab_Date	date	3
@New_Blood_Lead_Result	numeric(4,1)	5
@New_Hemoglobin_Value	numeric(4,1)	5
@New_Lab_ID	int	4
@New_Blood_Test_Costs	money	8
@New_Sample_Type_ID	tinyint	1
@New_Taken_After_Property_Remediation_Completed	bit	1
@New_Exclude_Result	bit	1
@New_Client_Status_ID	smallint	2
@New_Notes	varchar(3000)	3000
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20130618
-- Description: Stored Procedure to update
--               blood test results records
-- =====
-- DROP PROCEDURE usp_upBloodTestResults
```

```
CREATE PROCEDURE [dbo].[usp_upBloodTestResults]
    -- Add the parameters for the stored procedure here

    @BloodTestResultsID int = NULL,
    @New_Sample_Date date = NULL,
    @New_Lab_Date date = NULL,
    @New_Blood_Lead_Result numeric(4,1) = NULL,
    -- @New_Flag smallint = NULL,

    @New_Hemoglobin_Value numeric(4,1) = NULL,
    @New_Lab_ID int = NULL,
    @New_Blood_Test_Costs money = NULL,
    @New_Sample_Type_ID tinyint = NULL,
    @New_Taken_After_Property_Remediation_Completed bit = NULL,
    @New_Exclude_Result bit = NULL,
    @New_Client_Status_ID smallint = NULL,
    @New_Notes varchar(3000) = NULL,
    @DEBUG BIT = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int, @NotesID int, @spupdateBloodTestResultssqlStr
NVARCHAR(4000);

    -- insert statements for procedure here

    BEGIN TRY
        -- Check if BloodTestResultsID is valid, if not return

        IF NOT EXISTS (SELECT BloodTestResultsID from BloodTestResults where BloodTest-
ResultsID = @BloodTestResultsID)
            BEGIN
                RAISERROR(15000, -1,-1,'usp_upBloodTestResults');
            END

        -- BUILD update statement

        if (@New_Blood_Lead_Result is null)
            select @New_Blood_Lead_Result = LeadValue from BloodTestResults where Blood-
TestResultsID = @BloodTestResultsID

        SELECT @spupdateBloodTestResultssqlStr = N'update BloodTestResults set Lead-
Value = @Blood_Lead_Result'

        IF (@New_Sample_Date IS NOT NULL)
            SELECT @spupdateBloodTestResultssqlStr = @spupdateBloodTestResultssqlStr +
N', SampleDate = @Sample_Date'
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_upBloodTestResults

```

    IF (@New_Lab_Date IS NOT NULL)
        SELECT @spupdateBloodTestResultssqlStr = @spupdateBloodTestResultssqlStr +
N', LabSubmissionDate = @Lab_Date'

    IF (@New_Hemoglobin_Value IS NOT NULL)
        SELECT @spupdateBloodTestResultssqlStr = @spupdateBloodTestResultssqlStr +
N', HemoglobinValue = @Hemoglobin_Value'

    IF (@New_Lab_ID IS NOT NULL)
        SELECT @spupdateBloodTestResultssqlStr = @spupdateBloodTestResultssqlStr +
N', LabID = @Lab_ID'

    IF (@New_Blood_Test_Costs IS NOT NULL)
        SELECT @spupdateBloodTestResultssqlStr = @spupdateBloodTestResultssqlStr +
N', BloodTestCosts = @Blood_Test_Costs'

    IF (@New_Sample_Type_ID IS NOT NULL)
        SELECT @spupdateBloodTestResultssqlStr = @spupdateBloodTestResultssqlStr +
N', SampleTypeID = @Sample_Type_ID'

    IF (@New_Taken_After_Property_Remediation_Completed IS NOT NULL)
        SELECT @spupdateBloodTestResultssqlStr = @spupdateBloodTestResultssqlStr +
N', TakenAfterPropertyRemediationCompleted = @Taken_After_Property_Remediation_-Completed'

    IF (@New_Exclude_Result IS NOT NULL)
        SELECT @spupdateBloodTestResultssqlStr = @spupdateBloodTestResultssqlStr +
N', ExcludeResult = @Exclude_Result'

    IF (@New_Client_Status_ID IS NOT NULL)
        SELECT @spupdateBloodTestResultssqlStr = @spupdateBloodTestResultssqlStr +
N', ClientStatusID = @Client_Status_ID'

-- make sure to only update record for specified BloodTestResults

    SELECT @spupdateBloodTestResultssqlStr = @spupdateBloodTestResultssqlStr + N'
WHERE BloodTestResultsID = @BloodTestResultsID'

    IF (@DEBUG = 1)
        SELECT @spupdateBloodTestResultssqlStr, LeadValue = @New_Blood_Lead_Result,
SampleDate = @New_Sample_Date, LabSubmissionDate = @New_Lab_Date
            , HemoglobinVlaue = @New_Hemoglobin_Value, LabID = @New_Lab_ID,
BloodTestCosts = @New_Blood_Test_Costs, SampleTypeID = @New_Sample_Type_ID
            , TakenAfterPropertyRemediationCompleted = @New_Taken_After_-Property_Remediation_Completed,
ExcludeResult = @New_Exclude_Result
            , ClientStatusID = @New_Client_Status_ID, BloodTestResultsID =
@BloodTestResultsID

        EXEC [sp_executesql] @spupdateBloodTestResultssqlStr
            , N'@Blood_Lead_Result numeric(4,1), @Sample_Date date, @Lab_Date date,
@Hemoglobin_Value numeric(4,1), @Lab_ID int, @Blood_Test_Costs money
            , @Sample_Type_ID tinyint, @Taken_After_Property_Remediation_Completed
bit, @Exclude_Result bit
            , @Client_Status_ID smallint, @BloodTestResultsID int'
            , @Blood_Lead_Result = @New_Blood_Lead_Result
            , @Sample_Date = @New_Sample_Date
            , @Lab_Date = @New_Lab_Date

```

Author: liam

```
, @Hemoglobin_Value = @New_Hemoglobin_Value
, @Lab_ID = @New_Lab_ID
, @Blood_Test_Costs = @New_Blood_Test_Costs
, @Sample_Type_ID = @New_Sample_Type_ID
, @Taken_After_Property_Remediation_Completed = @New_Taken_After_Property_Remediation_Completed
, @Exclude_Result = @New_Exclude_Result
, @Client_Status_ID = @New_Client_Status_ID
, @BloodTestResultsID = @BloodTestResultsID

IF (@New_Notes IS NOT NULL)
EXEC      [dbo].[usp_InsertBloodTestResultsNotes]
          @BloodTestResults_ID = @BloodTestResultsID,
          @Notes = @New_Notes,
          @InsertedNotesID = @NotesID OUTPUT

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[BloodTestResults]
[dbo].[usp_InsertBloodTestResultsNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_upBloodTestResultsWebScreen]

[dbo].[usp_upBloodTestResultsWebScreen]	

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@BloodTestResultsID	int	4
@New_Sample_Date	date	3
@New_Lab_Date	date	3
@New_Blood_Lead_Result	numeric(4,1)	5
@New_Sample_Type_ID	tinyint	1
@New_Lab_ID	int	4
@New_Flag	smallint	2
@New_Client_Status_ID	smallint	2
@New_Hemoglobin_Value	numeric(4,1)	5
@New_Blood_Test_Costs	money	8
@New_Taken_After_Property_Remediation_Completed	bit	1
@New_Exclude_Result	bit	1
@New_Notes	varchar(3000)	3000
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150618
-- Description: stored procedure to update blood test results
--               data
-- =====
```

Author: liam

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_upBloodTestResultsWebScreen

```
CREATE PROCEDURE [dbo].[usp_upBloodTestResultsWebScreen]
    -- Add the parameters for the stored procedure here

    @BloodTestResultsID int = NULL,
    @New_Sample_Date date = NULL,
    @New_Lab_Date date = NULL,
    @New_Blood_Lead_Result numeric(4,1) = NULL,
    @New_Sample_Type_ID tinyint = NULL,
    @New_Lab_ID int = NULL,
    @New_Flag smallint = NULL,
    @New_Client_Status_ID smallint = NULL,
    @New_Hemoglobin_Value numeric(4,1) = NULL,
    @New_Blood_Test_Costs money = NULL,
    @New_Taken_After_Property_Remediation_Completed bit = NULL,
    @New_Exclude_Result bit = NULL,
    @New_Notes varchar(3000) = NULL,
    @DEBUG BIT = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    BEGIN
        DECLARE @ErrorLogID int, @RetestDate_return_value int,
                @updateBloodTestResultsReturnValue int;

        -- If no family ID was passed in exit

        IF (@BloodTestResultsID IS NULL)
        BEGIN
            RAISERROR ('Blood test results ID must be supplied', 11, -1);
            RETURN;
        END;

        BEGIN TRY
            -- update person flag/retest date

            IF (@New_Flag IS NOT NULL)
            BEGIN
                declare @Retest_Date date, @Sample_Date date, @Person_ID int;

                -- determine personID

                select @Person_ID = PersonID, @Sample_Date = SampleDate from BloodTest-
Results where BloodTestResultsID = @BloodTestResultsID

                -- set the retest date based on integer value passed in as Flag

                SET @Retest_Date = DATEADD(dd,@New_Flag,@Sample_Date);
            END;
        END TRY
        CATCH
        BEGIN
            IF @@TRANCOUNT > 0
                ROLLBACK TRANSACTION;
        END;
    END;
END;
```

Author: liam

```
-- update Person table with the new retest date

-- anyone with a blood test is a client

EXEC      @RetestDate_return_value = [dbo].[usp_upPerson]
          @Person_ID = @Person_ID
          , @New_RetestDate = @Retest_Date
          , @New_ClientStatusID = @New_Client_Status_ID;
END

-- update bloodtestResults

EXEC      @updateBloodTestResultsReturnValue = [dbo].[usp_upBloodTestResults]
          @BloodTestResultsID = @Blood-
TestResultsID,
          @New_Sample_Date = @New_Sample_-
Date,
          @New_Lab_Date = @New_Lab_Date,
          @New_Blood_Lead_Result = @New_-
Blood_Lead_Result,
          @New_Hemoglobin_Value = @New_-
Hemoglobin_Value,
          @New_Lab_ID = @New_Lab_ID,
          @New_Blood_Test_Costs = @New_-
Blood_Test_Costs,
          @New_Sample_Type_ID = @New_-
Sample_Type_ID,
          @New_Taken_After_Property_-
Remediation_Completed = @New_Taken_After_Property_Remediation_Completed,
          @New_Exclude_Result = @New_-
Exclude_Result,
          @New_Client_Status_ID = @New_-
Client_Status_ID,
          @New_Notes = @New_Notes,
          @DEBUG = @DEBUG

END TRY
BEGIN CATCH -- insert person

-- Call procedure to print error information.

EXECUTE dbo.uspPrintError;

-- Roll back any active or uncommittable transactions before

-- inserting information in the ErrorLog.

IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
```

```
    END CATCH; -- insert new person

    END
END

GO
```

Uses

[dbo].[BloodTestResults]
[dbo].[usp_upBloodTestResults]
[dbo].[usp_upPerson]
[dbo].[uspLogError]
[dbo].[uspPrintError]

[dbo].[usp_upClientFlag]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150605
-- Description: procedure to update the isClient
--               flag to 1 if the person has completed
--               a bloodtest or a questionnaire.
-- =====

CREATE PROCEDURE [dbo].[usp_upClientFlag]
    -- Add the parameters for the stored procedure here
    @DEBUG bit = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @spexecutesqlStr NVARCHAR(4000), @Recompile BIT = 1, @ErrorLogID int;

```

```

BEGIN TRY
    -- Set isClient true if person has a bloodtest or questionnaire

    update Person Set isClient = 1 where isClient = 0 AND PersonID IN
    ( Select PersonID from BloodTestResults
        UNION
        Select PersonID from Questionnaire
    )

    -- Set isClient false if person does not have a bloodtest or a questionnaire

    update Person Set isClient = 0 where isClient = 1 AND PersonID NOT IN
    ( Select PersonID from BloodTestResults
        UNION
        Select PersonID from Questionnaire
    )

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO

```

Uses

[dbo].[BloodTestResults]
 [dbo].[Person]
 [dbo].[Questionnaire]
 [dbo].[usp.LogError]
 [dbo].[uspPrintError]

[dbo].[usp_upClientWebScreen]	

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Family_ID	int	4
@Person_ID	int	4
@New_FirstName	varchar(50)	50
@New_MiddleName	varchar(50)	50
@New_LastName	varchar(50)	50
@New_BirthDate	date	3
@New_Gender	char	1
@New_StatusID	smallint	2
@New_ForeignTravel	bit	1
@New_OutofSite	bit	1
@New_EatsForeignFood	bit	1
@New_EmailAddress	varchar(320)	320
@New_RetestDate	date	3
@New_Moved	bit	1
@New_MovedDate	date	3
@New_isClosed	bit	1
@New_isResolved	bit	1
@New_ClientNotes	varchar(3000)	3000
@New_TravelNotes	varchar(3000)	3000
@New_HobbyNotes	varchar(3000)	3000
@New_ReleaseNotes	varchar(3000)	3000
@New_GuardianID	int	4
@New_PersonCode	smallint	2
@New_isSmoker	bit	1
@New_isClient	bit	1
@New_NursingMother	bit	1
@New_NursingInfant	bit	1
@New_Pregnant	bit	1

@New_EthnicityID	tinyint	1
@New_LanguageID	tinyint	1
@New_PrimaryLanguage	bit	1
@New_HobbyID	int	4
@New_OccupationID	int	4
@New_Occupation_StartDate	date	3
@New_Occupation_EndDate	date	3
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150325
-- Description: stored procedure to update data
--               from the Add a new client web page
-- =====

CREATE PROCEDURE [dbo].[usp_upClientWebScreen]
    -- Add the parameters for the stored procedure here

    @Family_ID int = NULL,
    @Person_ID int = NULL,
    @New_FirstName varchar(50) = NULL,
    @New_MiddleName varchar(50) = NULL,
    @New_LastName varchar(50) = NULL,
    @New_BirthDate date = NULL,
    @New_Gender char(1) = NULL,
    @New_StatusID smallint = NULL,
    @New_ForeignTravel bit = NULL,
    @New_OutofSite bit = NULL,
    @New_EatsForeignFood bit = NULL,
    @New_EmailAddress varchar(320) = NULL,
    @New_RetestDate date = NULL,
    @New_Moved bit = NULL,
    @New_MovedDate date = NULL,
    @New_isClosed bit = 0,
    @New_isResolved bit = 0,
    @New_ClientNotes varchar(3000) = NULL,
    @New_TravelNotes varchar(3000) = NULL,
    @New_HobbyNotes varchar(3000) = NULL,
    @New_ReleaseNotes varchar(3000) = NULL,
```

```
@New_GuardianID int = NULL,
@New_PersonCode smallint = NULL,
@New_isSmoker bit = NULL,
@New_isClient bit = NULL,
@New_NursingMother bit = NULL,
@New_NursingInfant bit = NULL,
@New_Pregnant bit = NULL,
--@New_isNursing bit = NULL,

--@New_isPregnant bit = NULL,

@New_EthnicityID tinyint = NULL,
@New_LanguageID tinyint = NULL,
@New_PrimaryLanguage bit = 1,
@New_HobbyID int = NULL,
@New_OccupationID int = NULL,
@New_Occupation_StartDate date = NULL,
@New_Occupation_EndDate date = NULL,
@DEBUG BIT = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    BEGIN
        DECLARE @ErrorLogID int,
            @updatePerson_return_value int,
            @Ethnicity_return_value int,
            @PersontoFamily_return_value int,
            @PersontoLanguage_return_value int,
            @PersontoHobby_return_value int,
            @PersontoOccupation_return_value int,
            @PersontoEthnicity_return_value int;

        -- If no family ID was passed in exit

        IF (@Family_ID IS NULL OR @Person_ID IS NULL)
        BEGIN
            RAISERROR ('Family and Person must be supplied', 11, -1);
            RETURN;
        END;

        if (@New_LastName is null)
        BEGIN
            select @New_LastName = Lastname from Family where FamilyID = @Family_ID
        END

        BEGIN TRY -- update person
```

```
EXEC    @updatePerson_return_value = [dbo].[usp_upPerson]
        @Person_ID = @Person_ID,
        @New_FirstName = @New_FirstName,
        @New_MiddleName = @New_MiddleName,
        @New_LastName = @New_LastName,
        @New_BirthDate = @New_BirthDate,
        @New_Gender = @New_Gender,
        @New_StatusID = @New_StatusID,
        @New_ForeignTravel = @New_ForeignTravel,
        @New_OutofSite = @New_OutofSite,
        @New_EatsForeignFood = @New_EatsForeignFood,
        @New_EmailAddress = @New_EmailAddress,
        @New_RetestDate = @New_RetestDate,
        @New_Moved = @New_Moved,
        @New_MovedDate = @New_MovedDate,
        @New_isClosed = @New_isClosed,
        @New_isResolved = @New_isResolved,
        @New_PersonNotes = @New_ClientNotes,
        @New_HobbyNotes = @New_HobbyNotes,
        @New_TravelNotes = @New_TravelNotes,
        @New_ReleaseNotes = @New_ReleaseNotes,
        @New_GuardianID = @New_GuardianID,
        @New_PersonCode = @New_PersonCode,
        @New_isSmoker = @New_isSmoker,
        @New_isClient = @New_isClient,
        @New_NursingMother = @New_NursingMother,
        @New_NursingInfant = @New_NursingInfant,
        @New_Pregnant = @New_Pregnant,
        @DEBUG = @DEBUG

-- Associate person to Ethnicity

IF ((@New_EthnicityID IS NOT NULL) AND
    (NOT EXISTS (SELECT PersonID from PersontoEthnicity where Ethnicity-
ID = @New_EthnicityID and PersonID = @Person_ID)))
    EXEC    @Ethnicity_return_value = [dbo].[usp_InsertPersontoEthnicity]
            @PersonID = @Person_ID,
            @EthnicityID = @New_EthnicityID
-- CODE FOR FUTURE EXTENSIBILITY OF UPDATING ETHNICITY

--IF (@New_Ethnicity IS NOT NULL)

--EXEC    @Ethnicity_return_value = [dbo].[usp_upEthnicity]

--    @PersonID = @Person_ID,
--    @New_EthnicityID = @New_EthnicityID,
--    @DEBUG = @DEBUG,
--    @PersontoEthnicityID = @New_PersontoEthnicityID OUTPUT
```

```
-- Associate person to family

-- If the person isn't already associated with that family

    if NOT EXISTS (SELECT PersonID from PersontoFamily where FamilyID = @Family_ID and PersonID = @Person_ID)
        EXEC      @PersontoFamily_return_value = usp_InsertPersontoFamily
                    @PersonID = @Person_ID, @FamilyID = @Family_ID, @OUTPUT = @PersontoFamily_return_value OUTPUT;

-- Associate person to language

    IF (@New_LanguageID is not NULL)
        EXEC      @PersontoLanguage_return_value = usp_InsertPersontoLanguage
                    @LanguageID = @New_LanguageID, @PersonID = @Person_ID, @isPrimaryLanguage = @New_PrimaryLanguage;

-- associate person to Hobby

    IF ((@New_HobbyID is not NULL) AND
        (NOT EXISTS (SELECT PersonID from PersontoHobby where HobbyID = @New_HobbyID and PersonID = @Person_ID)))
        EXEC      @PersontoHobby_return_value = usp_InsertPersontoHobby
                    @HobbyID = @New_HobbyID, @PersonID = @Person_ID;

-- associate person to occupation

    if ((@New_OccupationID is not NULL))
        IF (NOT EXISTS (SELECT PersonID from PersontoOccupation where OccupationID = @New_OccupationID and PersonID = @Person_ID))
            EXEC      @PersontoOccupation_return_value = [dbo].[usp_InsertPersontoOccupation]
                        @PersonID = @Person_ID,
                        @OccupationID = @New_OccupationID,
                        @StartDate = @New_Occupation_StartDate,
                        @EndDate = @New_Occupation_EndDate
        ELSE
            EXEC      @PersontoOccupation_return_value = [dbo].[usp_upOccupation]
                        @PersonID = @Person_ID,
                        @OccupationID = @New_OccupationID,
                        @Occupation_StartDate = @New_Occupation_StartDate,
                        @Occupation_EndDate = @New_Occupation_EndDate,
                        @DEBUG = @DEBUG;
    END TRY
    BEGIN CATCH -- insert person

        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.
```

```
IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH; -- insert new person

END
END

GO
```

Uses

[dbo].[Family]
[dbo].[PersontoEthnicity]
[dbo].[PersontoFamily]
[dbo].[PersontoHobby]
[dbo].[PersontoOccupation]
[dbo].[usp_InsertPersontoEthnicity]
[dbo].[usp_InsertPersontoFamily]
[dbo].[usp_InsertPersontoHobby]
[dbo].[usp_InsertPersontoLanguage]
[dbo].[usp_InsertPersontoOccupation]
[dbo].[usp_upOccupation]
[dbo].[usp_upPerson]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_upFamily]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Family_ID	int	4
@New_Last_Name	varchar(50)	50
@New_Number_of_Smokers	tinyint	1
@New_Primary_Language_ID	tinyint	1
@New_Notes	varchar(3000)	3000
@New_Pets	tinyint	1
@New_Frequently_Wash_Pets	bit	1
@New_Pets_in_and_out	bit	1
@New_Primary_Property_ID	int	4
@New_ForeignTravel	bit	1
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20150214
-- Description: Stored Procedure to update Family information
-- =====

CREATE PROCEDURE [dbo].[usp_upFamily]
    -- Add the parameters for the stored procedure here

    @Family_ID int = NULL,
    @New_Last_Name varchar(50) = NULL,
    @New_Number_of_Smokers tinyint = 0,
```

```

@New_Primary_Language_ID tinyint = 1,
@New_Notes varchar(3000) = NULL,
@New_Pets tinyint = NULL,
@New_Frequently_Wash_Pets bit = NULL,
@New_Pets_in_and_out bit = NULL,
@New_Primary_Property_ID int = NULL,
@New_ForeignTravel bit = NULL,
@DEBUG BIT = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    -- Insert statements for procedure here

    DECLARE @ErrorLogID int, @spupdateFamilySqlStr nvarchar(4000)
        , @NotesID INT, @Recompile BIT = 1;

    BEGIN TRY -- update Family information

        -- BUILD update statement

        IF (@New_Last_Name IS NULL)
            SELECT @New_Last_Name = LastName from family where FamilyID = @Family_ID;

        SELECT @spupdateFamilySqlStr = N'update Family set Lastname = @LastName'

        IF (@New_Number_of_Smokers IS NOT NULL)
            SELECT @spupdateFamilySqlStr = @spupdateFamilySqlStr + N', NumberofSmokers = @NumberofSmokers'

        IF (@New_Primary_Language_ID IS NOT NULL)
            SELECT @spupdateFamilySqlStr = @spupdateFamilySqlStr + N', PrimaryLanguage-ID = @PrimaryLanguageID'

        IF (@New_Pets IS NOT NULL)
            SELECT @spupdateFamilySqlStr = @spupdateFamilySqlStr + N', Pets = @Pets'

        IF (@New_Frequently_Wash_Pets IS NOT NULL)
            SELECT @spupdateFamilySqlStr = @spupdateFamilySqlStr + N', FrequentlyWash-Pets = @FrequentlyWashPets'

        IF (@New_Pets_in_and_out IS NOT NULL)
            SELECT @spupdateFamilySqlStr = @spupdateFamilySqlStr + N', Petsinandout = @Petsinandout'

        IF (@New_Primary_Property_ID IS NOT NULL)
            SELECT @spupdateFamilySqlStr = @spupdateFamilySqlStr + N', PrimaryProperty-ID = @PrimaryPropertyID'

```

```

IF (@New_ForeignTravel IS NOT NULL)
    SELECT @spupdateFamilySqlStr = @spupdateFamilySqlStr + N', ForeignTravel =
@ForeignTravel'

SELECT @spupdateFamilySqlStr = @spupdateFamilySqlStr + N' WHERE FamilyID =
@FamilyID'

IF @DEBUG = 1
    SELECT @spupdateFamilySqlStr, 'Lastname' = @New_Last_Name, 'Numberof-
Smokers' = @New_Number_of_Smokers
        , 'PrimaryLanguageID' = @New_Primary_Language_ID, 'Pets' = @New_Pets,
'Petsonandout' = @New_Pets_in_and_out
        , 'PrimaryPropertyID' = @New_Primary_Property_ID, 'FrequentlyWashPets' =
@New_Frequently_Wash_Pets
        , 'ForeignTravel' = @New_ForeignTravel

IF (@New_Notes IS NOT NULL)
BEGIN
    IF @DEBUG = 1
        SELECT 'EXEC [dbo].[usp_InsertFamilyNotes] @Family_ID = @Family_ID,
@Notes = @New_Notes, @InsertedNotesID = @NotesID OUTPUT '
            , @Family_ID, @New_Notes

        EXEC      [dbo].[usp_InsertFamilyNotes]
                    @Family_ID = @Family_ID,
                    @Notes = @New_Notes,
                    @InsertedNotesID = @NotesID OUTPUT
    END

-- update Family table

EXEC [sp_executesql] @spupdateFamilySqlStr
    , N'@LastName VARCHAR(50), @NumberofSmokers tinyint, @PrimaryLanguageID
tinyint
        , @Pets tinyint, @Petsonandout BIT, @PrimaryPropertyID int, @Frequently-
WashPets bit, @ForeignTravel bit, @FamilyID int'
        , @LastName = @New_Last_Name
        , @NumberofSmokers = @New_Number_of_Smokers
        , @PrimaryLanguageID = @New_Primary_Language_ID
        , @Pets = @New_Pets
        , @Petsonandout = @New_Pets_in_and_out
        , @PrimaryPropertyID = @New_Primary_Property_ID
        , @FrequentlyWashPets = @New_Frequently_Wash_Pets
        , @ForeignTravel = @New_ForeignTravel
        , @FamilyID = @Family_ID
END TRY -- update Family

BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before

```

```
-- inserting information in the ErrorLog.

IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Family]
[dbo].[usp_InsertFamilyNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_upFamilyWebScreen]

[dbo].[usp_upFamilytoProperty]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@FamilytoPropertyID	int	4
@PropertyLinkTypeID	int	4
@StartDate	date	3
@EndDate	date	3
@isPrimaryResidence	bit	1
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20150417
-- Description: Stored Procedure to update new FamilytoProperty records
-- =====

CREATE PROCEDURE [dbo].[usp_upFamilytoProperty]
    -- Add the parameters for the stored procedure here

    @FamilytoPropertyID int = NULL,
    @PropertyLinkTypeID int = NULL,
    @StartDate date = NULL,
    @EndDate date = NULL,
    @isPrimaryResidence bit = NULL,
    @DEBUG bit = 0

AS
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_upFamilytoProperty

```
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int, @spexecutesQLStr nvarchar(4000);
    -- Insert statements for procedure here

    BEGIN TRY
        SELECT @spexecutesQLStr = N'update FamilytoProperty set EndDate = @End_Date';

        IF @PropertyLinkTypeID IS NOT NULL
            SELECT @spexecutesQLStr = @spexecutesQLStr + ', PropertyLinkTypeID = @Property_Link_Type_ID';

        IF @StartDate IS NOT NULL
            SELECT @spexecutesQLStr = @spexecutesQLStr + ', StartDate = @Start_Date';

        IF @isPrimaryResidence IS NOT NULL
            SELECT @spexecutesQLStr = @spexecutesQLStr + ', isPrimaryResidence = @is_Primary_Residence';

        -- Add filters to update the correct record

        SELECT @spexecutesQLStr = @spexecutesQLStr + ' Where FamilytoPropertyID =
@Family_to_Property_ID';

        IF (@DEBUG = 1)
            SELECT @spexecutesQLStr, 'FamilytoPropertyID' = @FamilytoPropertyID,
'PropertyLinkTypeID' = @PropertyLinkTypeID
            , 'StartDate' = @StartDate, 'EndDate' = @EndDate, 'isPrimaryResidence'
= @isPrimaryResidence;

        EXEC [sp_executesql] @spexecutesQLStr
            , N'@Family_to_Property_ID int, @Property_Link_Type_ID int, @Start_Date
date, @End_Date date, @is_Primary_Residence bit'
            , @Family_to_Property_ID = @FamilytoPropertyID
            , @Property_Link_Type_ID = @PropertyLinkTypeID
            , @Start_Date = @StartDate
            , @End_Date = @EndDate
            , @is_Primary_Residence = @isPrimaryResidence;

    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.
```

Author: liam

```
IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END

GO
```

Uses

[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertFamilytoProperty]

[dbo].[usp_upFamilyWebScreen]	
-------------------------------	--

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Family_ID	int	4
@isNewAddress	bit	1
@New_Last_Name	varchar(50)	50
@PropertyID	int	4
@New_ConstructionType	int	4
@New_ArealD	int	4
@New_isinHistoricDistrict	bit	1
@New_isRemodeled	bit	1
@New_RemodelDate	date	3
@New_isinCityLimits	bit	1
@New_Address_Line1	varchar(100)	100
@New_Address_Line2	varchar(100)	100
@New_CityName	varchar(50)	50
@New_County	varchar(50)	50
@New_StateAbbr	char(2)	2
@New_ZipCode	varchar(10)	10
@New_Year_Built	date	3
@New_PropertyLinkTypeID	tinyint	1
@New_Movein_Date	date	3
@New_MoveOut_Date	date	3
@New_isPrimaryResidence	bit	1
@New_Owner_id	int	4
@New_is_Owner_Occupied	bit	1
@New_ReplacedPipesFaucets	bit	1
@New_TotalRemediationCosts	money	8
@New_PropertyNotes	varchar(3000)	3000
@New_is_Residential	bit	1
@New_isCurrentlyBeingRemodeled	bit	1

@New_has_Peeling_Chipping_Patin	bit	1
@New_is_Rental	bit	1
@New_PrimaryPhone	bigint	8
@PrimaryPhonePriority	tinyint	1
@New_SecondaryPhone	bigint	8
@SecondaryPhonePriority	tinyint	1
@New_Number_of_Smokers	tinyint	1
@New_Primary_Language_ID	tinyint	1
@New_Family_Notes	varchar(3000)	3000
@New_Pets	tinyint	1
@New_Frequently_Wash_Pets	bit	1
@New_Pets_in_and_out	bit	1
@New_ForeignTravel	bit	1
@New_OwnerContactInformation	varchar(1000)	1000
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20150329
-- Description: Stored Procedure to update Family
--               web screen information
-- =====

CREATE PROCEDURE [dbo].[usp_upFamilyWebScreen]
    -- Add the parameters for the stored procedure here

    @Family_ID int = NULL,
    @isNewAddress bit = 0,
    @New_Last_Name varchar(50) = NULL,
    @PropertyID int = NULL,
    @New_ConstructionType int = NULL,
    @New_AreaID int = NULL,
    @New_isinHistoricDistrict bit = NULL,
    @New_isRemodeled bit = NULL,
    @New_RemodelDate date = NULL,
    @New_isinCityLimits bit = NULL,
    @New_Address_Line1 varchar(100) = NULL,
    @New_Address_Line2 varchar(100) = NULL,
```

```
@New_CityName varchar(50) = NULL,
@New_County varchar(50) = NULL,
@New_StateAbbr char(2) = NULL,
@New_ZipCode varchar(10) = NULL,
@New_Year_Built date = NULL,
@New_PropertyLinkTypeID tinyint = NULL,
@New_Movein_Date date = NULL,
@New_MoveOut_Date date = NULL,
@New_isPrimaryResidence bit = NULL,
@New_Owner_id int = NULL,
@New_is_Owner_Occupied bit = NULL,
@New_ReplacedPipesFaucets bit = NULL,
@New_TotalRemediationCosts money = NULL,
@New_PropertyNotes varchar(3000) = NULL,
@New_is_Residential bit = NULL,
@New_isCurrentlyBeingRemodeled bit = NULL,
@New_has_Peeling_Chipping_Patin bit = NULL,
@New_is_Rental bit = NULL,
@New_PrimaryPhone bigint = NULL,
@PrimaryPhonePriority tinyint = 1,
@New_SecondaryPhone bigint = NULL,
@SecondaryPhonePriority tinyint = 2,
@New_Number_of_Smokers tinyint = NULL,
@New_Primary_Language_ID tinyint = 1,
@New_Family_Notes varchar(3000) = NULL,
@New_Pets tinyint = NULL,
@New_Frequently_Wash_Pets bit = NULL,
@New_Pets_in_and_out bit = NULL,
-- @New_Primary_Property_ID int = NULL,

@New_ForeignTravel bit = NULL,
@New_OwnerContactInformation varchar(1000) = NULL,
@DEBUG BIT = 1

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    -- Insert statements for procedure here

    DECLARE @ErrorLogID int
        , @Update_Family_return_value int
        , @Update_Property_return_value int
        , @NotesID INT, @Recompile BIT = 1
        , @New_Primary_Property_ID int
        , @Primaryphone_return_value int
        , @Secondaryphone_return_value int
        , @FamilytoProperty_return_value int
        , @InsertedFamilytoPropertyID int
```

```
, @DebugLogID int

BEGIN TRY -- update Family information

    -- Exit if family isn't specified

    IF (@Family_ID IS NULL or @Family_ID = '')
    BEGIN
        RAISERROR ('Family must be specified', 11, 1);
        RETURN;
    END;

    -- Existing property need the primary property id

    IF (@isNewAddress = 0)
    BEGIN
        -- Select the primary property id

        Select @PropertyID = PrimaryPropertyID from Family where FamilyID =
@Family_ID;

        EXEC @Update_Property_return_value = [dbo].[usp_upProperty]
            @PropertyID = @PropertyID,
            @New_ConstructionTypeID = @New_ConstructionType,
            @New_AreaID = @New_AreaID,
            @New_isinHistoricDistrict = @New_isinHistoricDistrict,
            @New_isRemodeled = @New_isRemodeled,
            @New_RemodelDate = @New_RemodelDate,
            @New_isinCityLimits = @New_isinCityLimits,
            @New_AddressLine1 = @New_Address_Line1,
            @New_AddressLine2 = @New_Address_Line2,
            @New_City = @New_CityName,
            @New_State = @New_StateAbbr,
            @New_Zipcode = @New_ZipCode,
            @New_YearBuilt = @New_Year_Built,
            @New_Ownerid = @New_Owner_id,
            @New_isOwnerOccupied = @New_is_Owner_Occupied,
            @New_ReplacedPipesFaucets = @New_ReplacedPipesFaucets,
            @New_TotalRemediationCosts = @New_TotalRemediationCosts,
            @New_PropertyNotes = @New_PropertyNotes,
            @New_isResidential = @New_is_Residential,
            @New_isCurrentlyBeingRemodeled = @New_isCurrentlyBeingRemodeled,
            @New_hasPeelingChippingPaint = @New_has_Peeling_Chipping_Patin,
            @New_County = @New_County,
            @New_isRental = @New_is_Rental,
            @New_OwnerContactInformation = @New_OwnerContactInformation,
            @DEBUG = @DEBUG

        -- SET the new primary property ID

        SET @New_Primary_Property_ID = @PropertyID;
    END
```

```
IF (@isNewAddress = 1)
BEGIN
    EXEC [dbo].[usp_InsertProperty]
        @ConstructionTypeID = @New_ConstructionType,
        @AreaID = @New_AreaID,
        @isinHistoricDistrict = @New_isinHistoricDistrict,
        @isRemodeled = @New_isRemodeled,
        @RemodelDate = @New_RemodelDate,
        @isinCityLimits = @New_isinCityLimits,
        @AddressLine1 = @New_Address_Line1,
        @AddressLine2 = @New_Address_Line2,
        @City = @New_CityName,
        @State = @New_StateAbbr,
        @Zipcode = @New_ZipCode,
        @YearBuilt = @New_Year_Built,
        @Ownerid = @New_Owner_id,
        @isOwnerOccupied = @New_is_Owner_Occupied,
        @ReplacedPipesFaucets = @New_ReplacedPipesFaucets,
        @TotalRemediationCosts = @New_TotalRemediationCosts,
        @New_PropertyNotes = @New_PropertyNotes,
        -- @isResidential = @New_isResidential,

        @isCurrentlyBeingRemodeled = @New_isCurrentlyBeingRemodeled,
        @hasPeelingChippingPaint = @New_has_Peeling_Chipping_Patin,
        @County = @New_County,
        -- @isRental = @New_isRental,

        --@OverRideDuplicate = @New_OverRideDuplicate,

        @OwnerContactInformation = @New_OwnerContactInformation,
        @PropertyID = @New_Primary_Property_ID OUTPUT
END

EXEC @Update_Family_return_value = [dbo].[usp_upFamily]
    @Family_ID = @Family_ID,
    @New_Last_Name = @New_Last_Name,
    @New_Number_of_Smokers = @New_Number_of_Smokers,
    @New_Primary_Language_ID = @New_Primary_Language_ID,
    @New_Notes = @New_Family_Notes,
    @New_Pets = @New_Pets,
    @New_Frequently_Wash_Pets = @New_Frequently_Wash_Pets,
    @New_Pets_in_and_out = @New_Pets_in_and_out,
    @New_Primary_Property_ID = @New_Primary_Property_ID,
    @New_ForeignTravel = @New_ForeignTravel,
    @DEBUG = @DEBUG;

EXEC @FamilytoProperty_return_value = [usp_InsertFamilytoProperty]
    @FamilyID = @Family_ID,
    @PropertyID = @New_Primary_Property_ID,
    @PropertyLinkTypeID = @New_PropertyLinkTypeID,
    @StartDate = @New_Movein_Date,
    @EndDate = @New_MoveOut_Date,
    @isPrimaryResidence = @New_isPrimaryResidence,
```

```
        @NewFamilytoPropertyID = @InsertedFamilytoPropertyID OUTPUT

        if (@New_PrimaryPhone is not NULL)
        BEGIN -- insert Primary Phone

            DECLARE @PrimaryPhoneNumberID_OUTPUT bigint, @PhoneTypeID tinyint;

            SELECT @PhoneTypeID = PhoneNumberTypeID from PhoneNumberType where
PhoneNumberTypeName = 'Primary Phone';

            EXEC    @Primaryphone_return_value = [dbo].[usp_InsertPhoneNumber]
                    @PhoneNumber = @New_PrimaryPhone,
                    @PhoneNumberTypeID = @PhoneTypeID,
                    @DEBUG = @DEBUG,
                    @PhoneNumberID_OUTPUT = @PrimaryPhoneNumberID_OUTPUT OUTPUT

            EXEC    [dbo].[usp_InsertFamilytoPhoneNumber]
                    @FamilyID = @Family_ID,
                    @NumberPriority = @PrimaryPhonePriority,
                    @PhoneNumberID = @PrimaryPhoneNumberID_OUTPUT,
                    @DEBUG = @DEBUG
        END -- insert Primary Phone

        if (@New_SecondaryPhone is not NULL)
        BEGIN -- insert Secondary Phone

            DECLARE @SecondaryPhoneNumberID_OUTPUT bigint;

            SELECT @PhoneTypeID = PhoneNumberTypeID from PhoneNumberType where
PhoneNumberTypeName = 'Secondary Phone';

            EXEC    @Secondaryphone_return_value = [dbo].[usp_InsertPhoneNumber]
                    @PhoneNumber = @New_SecondaryPhone,
                    @PhoneNumberTypeID = @PhoneTypeID,
                    @DEBUG = @DEBUG,
                    @PhoneNumberID_OUTPUT = @SecondaryPhoneNumberID_OUTPUT OUTPUT

            EXEC    [dbo].[usp_InsertFamilytoPhoneNumber]
                    @FamilyID = @Family_ID,
                    @NumberPriority = @SecondaryPhonePriority,
                    @PhoneNumberID = @SecondaryPhoneNumberID_OUTPUT,
                    @DEBUG = @DEBUG
        END -- insert Secondary Phone

    END TRY -- update Family

    BEGIN CATCH
        -- Call procedure to print error information.
    
```

```
EXECUTE dbo.uspPrintError;

-- Roll back any active or uncommittable transactions before

-- inserting information in the ErrorLog.

IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH;
END
GO
```

Uses

[dbo].[Family]
[dbo].[PhoneNumberType]
[dbo].[usp_InsertFamilytoPhoneNumber]
[dbo].[usp_InsertFamilytoProperty]
[dbo].[usp_InsertPhoneNumber]
[dbo].[usp_InsertProperty]
[dbo].[usp_upFamily]
[dbo].[usp_upProperty]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

[dbo].[usp_upOccupation]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PersonID	int	4
@OccupationID	tinyint	1
@Occupation_StartDate	date	3
@Occupation_EndDate	date	3
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20150327
-- Description: Stored Procedure to update occupation records
-- =====
-- DROP PROCEDURE usp_upOccupation

CREATE PROCEDURE [dbo].[usp_upOccupation]
    -- Add the parameters for the stored procedure here

    @PersonID int = NULL,
    @OccupationID tinyint = NULL,
    @Occupation_StartDate date = NULL,
    @Occupation_EndDate date = NULL,
    @DEBUG BIT = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
```

Author: liam

```

-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int, @ErrorMessage NVARCHAR(4000), @Update bit, @spupdate-
OccupationsqlStr NVARCHAR(4000);

-- Assume there is nothing to update

SET @Update = 0;

-- insert statements for procedure here

BEGIN TRY
    IF (@PersonID IS NULL OR @OccupationID IS NULL)
        BEGIN
            RAISERROR('Occupation and Person must be specified',11,50000);
            RETURN;
        END

    IF (NOT EXISTS (SELECT PersonID from PersontoOccupation where PersonID =
@PersonID AND OccupationID = @OccupationID))
        BEGIN
            SELECT @ErrorMessage = 'Specified person: ' + cast(@PersonID as varchar) + '
is not associated with occupation: '
            + cast(@OccupationID as varchar) + '. Try creating the association with
usp_InsertPersontoOccupation';
            RAISERROR(@ErrorMessage,8,50000);
            RETURN;
        END

    -- BUILD update statement

    SELECT @spupdateOccupationsqlStr = N'update PersontoOccupation Set PersonID =
@PersonID'

    IF (@Occupation_StartDate IS NOT NULL)
        BEGIN
            SET @Update = 1;
            SELECT @spupdateOccupationsqlStr = @spupdateOccupationsqlStr + ', StartDate
= @StartDate'
        END

    IF (@Occupation_EndDate IS NOT NULL)
        BEGIN
            SET @Update = 1;
            SELECT @spupdateOccupationsqlStr = @spupdateOccupationsqlStr + ', ENDDate =
@ENDDate'
        END

    IF (@Update = 1)
        BEGIN
            SELECT @spupdateOccupationsqlStr = @spupdateOccupationsqlStr + ' WHERE
PersonID = @PersonID and OccupationID = @OccupationID'
        END

```

```

        IF  (@DEBUG = 1)
            SELECT @spupdateOccupationsqlStr, 'StartDate' = @Occupation_StartDate,
'EndDate' = @Occupation_EndDate,
            'PersonID' = @PersonID, 'OccupationID' = @OccupationID

        EXEC [sp_executesql] @spupdateOccupationsqlStr
            , N'@OccupationID tinyint, @PersonID int, @StartDate date, @EndDate
date'
            , @OccupationID = @OccupationID
            , @PersonID = @PersonID
            , @StartDate = @Occupation_StartDate
            , @EndDate = @Occupation_EndDate
        END
    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END

GO

```

Uses

[dbo].[PersontoOccupation]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_upClientWebScreen]

[dbo].[usp_upPerson]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Person_ID	int	4
@New_FirstName	varchar(50)	50
@New_MiddleName	varchar(50)	50
@New_LastName	varchar(50)	50
@New_BirthDate	date	3
@New_Gender	char	1
@New_StatusID	smallint	2
@New_ForeignTravel	bit	1
@New_OutofSite	bit	1
@New_EatsForeignFood	bit	1
@New_EmailAddress	varchar(320)	320
@New_RetestDate	date	3
@New_Moved	bit	1
@New_MovedDate	date	3
@New_isClosed	bit	1
@New_isResolved	bit	1
@New_PersonNotes	varchar(3000)	3000
@New_TravelNotes	varchar(3000)	3000
@New_HobbyNotes	varchar(3000)	3000
@New_ReleaseNotes	varchar(3000)	3000
@New_GuardianID	int	4
@New_PersonCode	smallint	2
@New_isSmoker	bit	1
@New_isClient	bit	1
@New_NursingMother	bit	1
@New_NursingInfant	bit	1
@New_Pregnant	bit	1
@New_ClientStatusID	smallint	2

@DEBUG	bit	1
--------	-----	---

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20130506
-- Description: Stored Procedure to update new people records
-- =====
-- DROP PROCEDURE usp_upPerson

CREATE PROCEDURE [dbo].[usp_upPerson]
    -- Add the parameters for the stored procedure here

    @Person_ID int = NULL,
    @New_FirstName varchar(50) = NULL,
    @New_MiddleName varchar(50) = NULL,
    @New_LastName varchar(50) = NULL,
    @New_BirthDate date = NULL,
    @New_Gender char(1) = NULL,
    @New_StatusID smallint = NULL,
    @New_ForeignTravel bit = NULL,
    @New_OutofSite bit = NULL,
    @New_EatsForeignFood bit = NULL,
    @New_EmailAddress varchar(320) = NULL,
    @New_RetestDate date = NULL,
    @New_Moved bit = NULL,
    @New_MovedDate date = NULL,
    @New_isClosed bit = 0,
    @New_isResolved bit = 0,
    @New_PersonNotes varchar(3000) = NULL,
    @New_TravelNotes varchar(3000) = NULL,
    @New_HobbyNotes varchar(3000) = NULL,
    @New_ReleaseNotes varchar(3000) = NULL,
    @New_GuardianID int = NULL,
    @New_PersonCode smallint = NULL,
    @New_isSmoker bit = NULL,
    @New_isClient bit = NULL,
    @New_NursingMother bit = NULL,
    @New_NursingInfant bit = NULL,
    @New_Pregnant bit = NULL,
    @New_ClientStatusID smallint = NULL,
    @DEBUG BIT = 0
AS
BEGIN
```

Author: liam

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_upPerson

```
-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @ErrorLogID int, @NotesID int, @spupdatePersonssqlStr NVARCHAR(4000);

-- insert statements for procedure here

BEGIN TRY
    -- Check if PersonID is valid, if not return

    IF NOT EXISTS (SELECT PersonID from Person where PersonID = @Person_ID)
    BEGIN
        RAISERROR(15000, -1,-1,'usp_upPerson');
    END

    -- BUILD update statement

    IF (@New_LastName IS NULL)
        SELECT @New_LastName = LastName from Person where PersonID = @Person_ID;

    SELECT @spupdatePersonssqlStr = N'update Person set Lastname = @LastName'

    IF (@New_FirstName IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', FirstName =
@Firstname'

    IF (@New_MiddleName IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', MiddleName =
@MiddleName'

    IF (@New_BirthDate IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', BirthDate =
@BirthDate'

    IF (@New_Gender IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', Gender =
@Gender'

    IF (@New_StatusID IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', StatusID =
@StatusID'

    IF (@New_ForeignTravel IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', ForeignTravel =
@ForeignTravel'

    IF (@New_OutofSite IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', OutofSite =
@OutofSite'

    IF (@New_EatsForeignFood IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', EatsForeignFood
```

Author: liam

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_upPerson

```

= @EatsForeignFood'

    IF (@New_EmailAddress IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', EmailAddress =
@EmailAddress'

    IF (@New_RetestDate IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', RetestDate =
@RetestDate'

    IF (@New_Moved IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', Moved = @Moved'

    IF (@New_MovedDate IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', MovedDate =
@MovedDate'

    IF (@New_isClosed IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', isClosed = @is-
Closed'

    IF (@New_isResolved IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', isResolved = @is-
Resolved'

    IF (@New_GuardianID IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', GuardianID =
@GuardianID'

    IF (@New_PersonCode IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', PersonCode =
@PersonCode'

    IF (@New_isSmoker IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', isSmoker = @is-
Smoker'

    IF (@New_isClient IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', isClient = @is-
Client'

    IF (@New_NursingMother IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', NursingMother =
@NursingMother'

    IF (@New_NursingInfant IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', NursingInfant =
@NursingInfant'

    IF (@New_Pregnant IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', Pregnant =
@Pregnant'

    IF (@New_ClientStatusID IS NOT NULL)
        SELECT @spupdatePersonssqlStr = @spupdatePersonssqlStr + N', ClientStatusID =
@ClientStatusID'

```

Author: liam

```

-- make sure to only update record for specified person

SELECT @spupdatePersonsSqlStr = @spupdatePersonsSqlStr + N' WHERE PersonID =
@PersonID'

IF (@DEBUG = 1)
    SELECT @spupdatePersonsSqlStr, LastName = @New_LastName, FirstName = @New_FirstName,
    MiddleName = @New_MiddleName
        , BirthDate = @New_BirthDate, Gender = @New_Gender, StatusID =
@New_StatusID, ForeignTravel = @New_ForeignTravel
        , OutofSite = @New_OutofSite, EatsForeignFood = @New_EatsForeign-
Food, EmailAddress = @New_EmailAddress, RetestDate = @New_RetestDate
        , Moved = @New_Moved, MovedDate = @New_MovedDate, isClosed =
@New_isClosed, isResolved = @New_isResolved
        , GuardianID = @New_GuardianID, PersonCode = @New_PersonCode, is-
Smoker = @New_isSmoker, isClient = @New_isClient
        , NursingMother = @New_NursingMother, NursingInfant = @New_Nursing-
Infant, Pregnant = @New_Pregnant
        , ClientStatusID = @New_ClientStatusID, PersonID = @Person_ID

EXEC [sp_executesql] @spupdatePersonsSqlStr
    , N'@LastName VARCHAR(50), @FirstName VARCHAR(50), @MiddleName
VARCHAR(50), @BirthDate date, @Gender char(1)
        , @StatusID smallint, @ForeignTravel BIT, @OutofSite bit, @EatsForeign-
Food bit, @EmailAddress varchar(320), @RetestDate date
        , @Moved bit, @MovedDate date, @isClosed bit, @isResolved bit,
@GuardianID int, @PersonCode smallint, @isSmoker bit
        , @isClient bit, @NursingMother bit, @NursingInfant bit, @Pregnant bit,
@ClientStatusID smallint, @PersonID int'
        , @LastName = @New_LastName
        , @FirstName = @New_FirstName
        , @MiddleName = @New_MiddleName
        , @BirthDate = @New_BirthDate
        , @Gender = @New_Gender
        , @StatusID = @New_StatusID
        , @ForeignTravel = @New_ForeignTravel
        , @OutofSite = @New_OutofSite
        , @EatsForeignFood = @New_EatsForeignFood
        , @EmailAddress = @New_EmailAddress
        , @RetestDate = @New_RetestDate
        , @Moved = @New_Moved
        , @MovedDate = @New_MovedDate
        , @isClosed = @New_isClosed
        , @isResolved = @New_isResolved
        , @GuardianID = @New_GuardianID
        , @PersonCode = @New_PersonCode
        , @isSmoker = @New_isSmoker
        , @isClient = @New_isClient
        , @NursingMother = @New_NursingMother
        , @NursingInfant = @New_NursingInfant
        , @Pregnant = @New_Pregnant
        , @ClientStatusID = @New_ClientStatusID
        , @PersonID = @Person_ID

IF (@New_PersonNotes IS NOT NULL)

```

```

EXEC      [dbo].[usp_InsertPersonNotes]
          @Person_ID = @Person_ID,
          @Notes = @New_PersonNotes,
          @InsertedNotesID = @NotesID OUTPUT

IF (@New_ReleaseNotes IS NOT NULL)
EXEC      [dbo].[usp_InsertPersonReleaseNotes]
          @Person_ID = @Person_ID,
          @Notes = @New_TravelNotes,
          @InsertedNotesID = @NotesID OUTPUT

IF (@New_TravelNotes IS NOT NULL)
EXEC      [dbo].[usp_InsertPersonTravelNotes]
          @Person_ID = @Person_ID,
          @Notes = @New_TravelNotes,
          @InsertedNotesID = @NotesID OUTPUT

IF (@New_HobbyNotes IS NOT NULL)
EXEC      [dbo].[usp_InsertPersonHobbyNotes]
          @Person_ID = @Person_ID,
          @Notes = @New_HobbyNotes,
          @InsertedNotesID = @NotesID OUTPUT

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END

    EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
    RETURN ERROR_NUMBER()
END CATCH;
END

GO

```

Uses

[dbo].[Person]
[dbo].[usp_InsertPersonHobbyNotes]
[dbo].[usp_InsertPersonNotes]
[dbo].[usp_InsertPersonReleaseNotes]
[dbo].[usp_InsertPersonTravelNotes]

[dbo].[usp.LogError]

[dbo].[uspPrintError]

Used By

[dbo].[usp_InsertNewBloodLeadTestResultsWebScreen]

[dbo].[usp_InsertNewQuestionnaireWebScreen]

[dbo].[usp_upBloodTestResultsWebScreen]

[dbo].[usp_upClientWebScreen]

[dbo].[usp_upProperty]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@PropertyID	int	4
@New_ConstructionTypeID	tinyint	1
@New_ArealD	int	4
@New_isinHistoricDistrict	bit	1
@New_isRemodeled	bit	1
@New_RemodelDate	date	3
@New_isinCityLimits	bit	1
@New_AddressLine1	varchar(100)	100
@New_AddressLine2	varchar(100)	100
@New_City	varchar(50)	50
@New_State	char(2)	2
@New_Zipcode	varchar(12)	12
@New_YearBuilt	date	3
@New_Ownerid	int	4
@New_isOwnerOccupied	bit	1
@New_ReplacedPipesFaucets	tinyint	1
@New_TotalRemediationCosts	money	8
@New_PropertyNotes	varchar(3000)	3000
@New_isResidential	bit	1
@New_isCurrentlyBeingRemodeled	bit	1
@New_hasPeelingChippingPaint	bit	1
@New_County	varchar(50)	50
@New_isRental	bit	1
@New_OwnerContactInformation	varchar(1000)	1000
@DEBUG	bit	1

SQL Script

```
-- =====
-- Author:      William Thier
-- Create date: 20140817
-- Description: Stored Procedure to update property records
-- =====

CREATE PROCEDURE [dbo].[usp_upProperty]    -- usp_upProperty
    -- Add the parameters for the stored procedure here

    @PropertyID int = NULL,
    @New_ConstructionTypeID tinyint = NULL,
    @New_AreaID int = NULL,
    @New_isinHistoricDistrict bit = NULL,
    @New_isRemodeled bit = NULL,
    @New_RemodelDate date = NULL,
    @New_isinCityLimits bit = NULL,
    @New_AddressLine1 varchar(100) = NULL,
    @New_AddressLine2 varchar(100) = NULL,
    @New_City varchar(50) = NULL,
    @New_State char(2) = NULL,
    @New_Zipcode varchar(12) = NULL,
    @New_YearBuilt date = NULL,
    @New_Ownerid int = NULL,
    @New_isOwnerOccupied bit = NULL,
    @New_ReplacedPipesFaucets tinyint = 0,
    @New_TotalRemediationCosts money = NULL,
    @New_PropertyNotes varchar(3000) = NULL,
    @New_isResidential bit = NULL,
    @New_isCurrentlyBeingRemodeled bit = NULL,
    @New_hasPeelingChippingPaint bit = NULL,
    @New_County varchar(50) = NULL,
    @New_isRental bit = NULL,
    @New_OwnerContactInformation varchar(1000) = NULL,
    @DEBUG bit = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;
```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_upProperty

```

DECLARE @ErrorLogID int, @NotesID int, @spupdatePropertyssqlStr nvarchar(4000);
-- Insert statements for procedure here

BEGIN TRY
    if (@PropertyID is NULL)
        BEGIN
            DECLARE @ErrorMessage VARCHAR(3000);
            SET @ErrorMessage = 'Property must be specified';
            RAISERROR (@ErrorMessage, 11, -1);
            RETURN;
        END
    -- BUILD update statement

    IF (@New_isinHistoricDistrict IS NULL)
        SELECT @New_isinHistoricDistrict = isinHistoricDistrict from Property where
PropertyID = @PropertyID;

    SELECT @spupdatePropertyssqlStr = N'update Property set isinHistoricDistrict =
@isinHistoricDistrict'

    IF (@New_ConstructionTypeID IS NOT NULL)
        SELECT @spupdatePropertyssqlStr = @spupdatePropertyssqlStr + N', Construction-
TypeID = @ConstructionTypeID'

    IF (@New_AreaID IS NOT NULL)
        SELECT @spupdatePropertyssqlStr = @spupdatePropertyssqlStr + N', AreaID =
@AreaID'

    IF (@New_isRemodeled IS NOT NULL)
        SELECT @spupdatePropertyssqlStr = @spupdatePropertyssqlStr + N', isRemodeled =
@isRemodeled'

    IF (@New_RemodelDate IS NOT NULL)
        SELECT @spupdatePropertyssqlStr = @spupdatePropertyssqlStr + N', RemodelDate =
@RemodelDate'

    IF (@New_isinCityLimits IS NOT NULL)
        SELECT @spupdatePropertyssqlStr = @spupdatePropertyssqlStr + N', isinCity-
Limits = @isinCityLimits'

    IF (@New_AddressLine1 IS NOT NULL)
        SELECT @spupdatePropertyssqlStr = @spupdatePropertyssqlStr + N', AddressLine1 =
@AddressLine1'

    IF (@New_AddressLine2 IS NOT NULL)
        SELECT @spupdatePropertyssqlStr = @spupdatePropertyssqlStr + N', AddressLine2 =
@AddressLine2'

    IF (@New_City IS NOT NULL)
        SELECT @spupdatePropertyssqlStr = @spupdatePropertyssqlStr + N', City =
@City'

    IF (@New_State IS NOT NULL)
        SELECT @spupdatePropertyssqlStr = @spupdatePropertyssqlStr + N', State =
@State'

```

Author: liam

```

    IF (@New_Zipcode IS NOT NULL)
        SELECT @spupdatePropertysqlStr = @spupdatePropertysqlStr + N', ZipCode =
@ZipCode'

    IF (@New_Ownerid IS NOT NULL)
        SELECT @spupdatePropertysqlStr = @spupdatePropertysqlStr + N', OwnerID =
@OwnerID'

    IF (@New_isOwnerOccupied IS NOT NULL)
        SELECT @spupdatePropertysqlStr = @spupdatePropertysqlStr + N', isOwner-
Occupied = @isOwnerOccupied'

    IF (@New_ReplacedPipesFaucets IS NOT NULL)
        SELECT @spupdatePropertysqlStr = @spupdatePropertysqlStr + N', Replaced-
PipesFaucets = @ReplacedPipesFaucets'

    IF (@New_TotalRemediationCosts IS NOT NULL)
        SELECT @spupdatePropertysqlStr = @spupdatePropertysqlStr + N', Total-
RemediationCosts = @TotalRemediationCosts'

    IF (@New_isResidential IS NOT NULL)
        SELECT @spupdatePropertysqlStr = @spupdatePropertysqlStr + N', is-
Residential = @isResidential'

    IF (@New_isCurrentlyBeingRemodeled IS NOT NULL)
        SELECT @spupdatePropertysqlStr = @spupdatePropertysqlStr + N', isCurrently-
BeingRemodeled = @isCurrentlyBeingRemodeled'

    IF (@New_hasPeelingChippingPaint IS NOT NULL)
        SELECT @spupdatePropertysqlStr = @spupdatePropertysqlStr + N', hasPeeling-
ChippingPaint = @hasPeelingChippingPaint'

    IF (@New_County IS NOT NULL)
        SELECT @spupdatePropertysqlStr = @spupdatePropertysqlStr + N', County =
@County'

    IF (@New_isRental IS NOT NULL)
        SELECT @spupdatePropertysqlStr = @spupdatePropertysqlStr + N', isRental =
@isRental'

    IF (@New_YearBuilt IS NOT NULL)
        SELECT @spupdatePropertysqlStr = @spupdatePropertysqlStr + N', YearBuilt =
@YearBuilt'

    IF (@New_OwnerContactInformation IS NOT NULL)
        SELECT @spupdatePropertysqlStr = @spupdatePropertysqlStr + N', OwnerContact-
Information = @OwnerContactInformation'

    SELECT @spupdatePropertysqlStr = @spupdatePropertysqlStr + N' WHERE PropertyID
= @PropertyID'

    -- update property table

    IF @DEBUG = 1
        SELECT @spupdatePropertysqlStr, New_ConstructionTypeID = @New_Construction-
TypeID, New_AreaID = @New_AreaID

```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_upProperty

```

        , New_isinHistoricDistrict = @New_isinHistoricDistrict, New_is-
Remodeled = @New_isRemodeled
        , New_RemodelDate = @New_RemodelDate, New_isinCityLimits =
@New_isinCityLimits
        , New_AddressLine1 = @New_AddressLine1, New_AddressLine2 = @New_-
AddressLine2, New_City = @New_City
        , New_State = @New_State, New_Zipcode = @New_Zipcode, New_OwnerID =
@New_Ownerid
        , New_isOwnerOccupied = @New_isOwnerOccupied, New_ReplacedPipes-
Faucets = @New_ReplacedPipesFaucets
        , New_PropertyNotes = @New_PropertyNotes, New_TotalRemediationCosts
= @New_TotalRemediationCosts
        , New_isResidential = @New_isResidential, New_isCurrentlyBeing-
Remodeled = @New_isCurrentlyBeingRemodeled
        , New_hasPeelingChippingPaint = @New_hasPeelingChippingPaint, New_-
County = @New_County
        , New_isRental = @New_isRental, New_YearBuilt = @New_YearBuilt
        , New_OwnerContactInformation = @New_OwnerContactInformation,
PropertyID = @PropertyID

    EXEC [sp_executesql] @spupdatePropertysqlStr
    , N'@ConstructionTypeID tinyint, @AreaID int, @isinHistoricDistrict bit,
@isRemodeled bit, @RemodelDate date
    , @isinCityLimits BIT, @AddressLine1 varchar(100), @AddressLine2
varchar(100), @City varchar(50), @State char(2)
    , @Zipcode varchar(12), @OwnerID int, @isOwnerOccupied bit, @ReplacedPipes-
Faucets tinyint, @TotalRemediationCosts money
    , @isResidential bit, @isCurrentlyBeingRemodeled bit, @hasPeelingChipping-
Paint bit
    , @County varchar(50), @isRental bit, @YearBuilt date, @OwnerContact-
Information varchar(1000), @PropertyID int'
    , @ConstructionTypeID = @New_ConstructionTypeID
    , @AreaID = @New_AreaID
    , @isinHistoricDistrict = @New_isinHistoricDistrict
    , @isRemodeled = @New_isRemodeled
    , @RemodelDate = @New_RemodelDate
    , @isinCityLimits = @New_isinCityLimits
    , @AddressLine1 = @New_AddressLine1
    , @AddressLine2 = @New_AddressLine2
    , @City = @New_City
    , @State = @New_State
    , @Zipcode = @New_Zipcode
    , @OwnerID = @New_Ownerid
    , @isOwnerOccupied = @New_isOwnerOccupied
    , @ReplacedPipesFaucets = @New_ReplacedPipesFaucets
    , @TotalRemediationCosts = @New_TotalRemediationCosts
    , @isResidential = @New_isResidential
    , @isCurrentlyBeingRemodeled = @New_isCurrentlyBeingRemodeled
    , @hasPeelingChippingPaint = @New_hasPeelingChippingPaint
    , @County = @New_County
    , @isRental = @New_isRental
    , @YearBuilt = @New_YearBuilt
    , @OwnerContactInformation = @New_OwnerContactInformation
    , @PropertyID = @PropertyID

    IF (@New_PropertyNotes IS NOT NULL)
    BEGIN

```

Author: liam

```

    IF @DEBUG = 1
        SELECT 'EXEC [dbo].[usp_InsertPropertyNotes] @Property_ID = @Property_-
ID, @Notes = @New_PropertyNotes, @InsertedNotesID = @NotesID OUTPUT '
            , @PropertyID, @New_PropertyNotes

        EXEC      [dbo].[usp_InsertPropertyNotes]
            @Property_ID = @PropertyID,
            @Notes = @New_PropertyNotes,
            @InsertedNotesID = @NotesID OUTPUT
    END

    END TRY
    BEGIN CATCH
        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;

        -- Roll back any active or uncommittable transactions before
        -- inserting information in the ErrorLog.

        IF XACT_STATE() <> 0
        BEGIN
            ROLLBACK TRANSACTION;
        END

        EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
        RETURN ERROR_NUMBER()
    END CATCH;
END

GO

```

Uses

[dbo].[Property]
[dbo].[usp_InsertPropertyNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_upFamilyWebScreen]

[dbo].[usp_upQuestionnaire]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@QuestionnaireID	int	4
@New_QuestionnaireDate	date	3
@New_QuestionnaireDataSourceID	int	4
@New_VisitRemodeledProperty	bit	1
@New_PaintDate	date	3
@New_RemodelPropertyDate	date	3
@New_isExposedtoPeelingPaint	bit	1
@New_isTakingVitamins	bit	1
@New_NursingMother	bit	1
@New_NursingInfant	bit	1
@New_Pregnant	bit	1
@New_isUsingPacifier	bit	1
@New_isUsingBottle	bit	1
@New_BitesNails	bit	1
@New_NonFoodEating	bit	1
@New_NonFoodinMouth	bit	1
@New_EatOutside	bit	1
@New_Suckling	bit	1
@New_Mouthing	bit	1
@New_FrequentHandWashing	bit	1
@New_VisitsOldHomes	bit	1
@New_DaycareID	int	4
@New_Notes	varchar(3000)	3000
@DEBUG	bit	1

SQL Script

Author: liam

```
-- =====

-- Author:      William Thier

-- Create date: 20130618

-- Description: Stored Procedure to update

--          questionnaire records

-- =====

-- DROP PROCEDURE usp_upQuestionnaire

CREATE PROCEDURE [dbo].[usp_upQuestionnaire]
    -- Add the parameters for the stored procedure here

    @QuestionnaireID int = NULL,
    @New_QuestionnaireDate date = NULL,
    @New_QuestionnaireDataSourceID int = NULL,
    @New_VisitRemodeledProperty bit = NULL,
    @New_PaintDate date = NULL,
    @New_RemodelPropertyDate date = NULL,
    @New_isExposedtoPeelingPaint bit = NULL,
    @New_isTakingVitamins bit = NULL,
    @New_NursingMother bit = NULL,
    @New_NursingInfant bit = NULL,
    @New_Pregnant bit = NULL,
    @New_isUsingPacifier bit = NULL,
    @New_isUsingBottle bit = NULL,
    @New_BitesNails bit = NULL,
    @New_NonFoodEating bit = NULL,
    @New_NonFoodinMouth bit = Null,
    @New_EatOutside bit = NULL,
    @New_Suckling bit = NULL,
    @New_Mouthing bit = NULL,
    @New_FrequentHandWashing bit = NULL,
    @New_VisitsOldHomes bit = NULL,
    @New_DaycareID int = NULL,
    @New_Notes varchar(3000) = NULL,
    @DEBUG BIT = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from

    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    DECLARE @ErrorLogID int, @NotesID int, @spupdateQuestionnairesqlStr NVARCHAR(4000);

    -- insert statements for procedure here
```

```

BEGIN TRY
    -- Check if QuestionnaireID is valid, if not return

    IF NOT EXISTS (SELECT QuestionnaireID from Questionnaire where QuestionnaireID = @QuestionnaireID)
        BEGIN
            RAISERROR('QuestionnaireID must be specified and valid', 11,-1,'usp_up-
Questionnaire');
        END

    -- BUILD update statement

    if (@New_QuestionnaireDate is null)
        select @New_QuestionnaireDate = QuestionnaireDate from Questionnaire where
QuestionnaireID = @QuestionnaireID

    SELECT @spupdateQuestionnairesqlStr = N'update Questionnaire set Questionnaire-
Date = @QuestionnaireDate'

    IF (@New_QuestionnaireDataSourceID IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N',',
QuestionnaireDataSourceID = @QuestionnaireDataSourceID'

    IF (@New_VisitRemodeledProperty IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N',',
VisitRemodeledProperty = @VisitRemodeledProperty'

    IF (@New_PaintDate IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N',',
PaintDate = @PaintDate'

    IF (@New_RemodelPropertyDate IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N',',
RemodelPropertyDate = @RemodelPropertyDate'

    IF (@New_isExposedtoPeelingPaint IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N',',
is-
ExposedtoPeelingPaint = @isExposedtoPeelingPaint'

    IF (@New_isTakingVitamins IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N',',
is-
TakingVitamins = @isTakingVitamins'

    IF (@New_NursingMother IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N',',
NursingMother = @NursingMother'

    IF (@New_NursingInfant IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N',',
NursingInfant = @NursingInfant'

    IF (@New_Pregnant IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N',',
Pregnant = @Pregnant'

```

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_upQuestionnaire

```

    IF (@New_isUsingPacifier IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N', is-
UsingPacifier = @isUsingPacifier'

    IF (@New_isUsingBottle IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N', is-
UsingBottle = @isUsingBottle'

    IF (@New_BitesNails IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N', 
BitesNails = @BitesNails'

    IF (@New_NonFoodEating IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N', 
NonFoodEating = @NonFoodEating'

    IF (@New_NonFoodinMouth IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N', 
NonFoodinMouth = @NonFoodinMouth'

    IF (@New_EatOutside IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N', 
EatOutside = @EatOutside'

    IF (@New_Suckling IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N', 
Suckling = @Suckling'

    IF (@New_Mouthing IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N', 
Mouthing = @Mouthing'

    IF (@New_FrequentHandWashing IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N', 
FrequentHandWashing = @FrequentHandWashing'

    IF (@New_VisitsOldHomes IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N', 
VisitsOldHomes = @VisitsOldHomes'

    IF (@New_DaycareID IS NOT NULL)
        SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N', 
DayCareID = @DaycareID'

-- make sure to only update record for specified Questionnaire

    SELECT @spupdateQuestionnairesqlStr = @spupdateQuestionnairesqlStr + N' WHERE
QuestionnaireID = @QuestionnaireID'

    IF (@DEBUG = 1)
        SELECT @spupdateQuestionnairesqlStr, QuestionnaireDate = @New_Questionnaire-
Date, QuestionnaireDataSourceID = @New_QuestionnaireDataSourceID
            , VisitRemodeledProperty = @New_VisitRemodeledProperty, PaintDate =
@New_PaintDate, RemodelPropertyDate = @New_RemodelPropertyDate
            , isExposedtoPeelingPaint = @New_isExposedtoPeelingPaint, isTaking-
Vitamins = @New_isTakingVitamins, NursingMother = @New_NursingMother
            , NursingInfant = @New_NursingInfant, Pregnant = @New_Pregnant, is-

```

Author: liam

Project> (local)> User databases> LCCHPDev> Programmability> Stored Procedures> dbo.usp_upQuestionnaire

```

UsingPacifier = @New_isUsingPacifier, isUsingBottle = @New_isUsingBottle
                , Bitesnails = @New_BitesNails, NonFoodEating = @New_NonFoodEating,
NonFoodinMouth = @New_NonFoodinMouth, EatOutside = @New_EatOutside
                , Suckling = @New_Suckling, Mouthing = @New_Mouthing, FrequentHand-
Washing = @New_FrequentHandWashing, VisitsOldHomes = @New_VisitsOldHomes
                , DaycareID = @New_DaycareID, QuestionnaireID = @QuestionnaireID,
DEBUG = @DEBUG

    EXEC [sp_executesql] @spupdateQuestionnairesqlStr
        , N'@QuestionnaireDate date, @QuestionnaireDataSourceID int, @Visit-
RemodeledProperty bit, @PaintDate date, @RemodelPropertyDate date
        , @isExposedtoPeelingPaint bit, @isTakingVitamins bit, @NursingMother
bit, @NursingInfant bit, @Pregnant bit, @isUsingPacifier bit
        , @isUsingBottle bit, @BitesNails bit, @NonFoodEating bit, @NonFoodin-
Mouth bit, @Eatoutside bit, @Suckling bit, @Mouthing bit
        , @FrequentHandWashing bit , @VisitsOldHomes bit, @DaycareID int,
@QuestionnaireID int'
        , @QuestionnaireDate = @New_QuestionnaireDate
        , @QuestionnaireDataSourceID = @New_QuestionnaireDataSourceID
        , @VisitRemodeledProperty = @New_VisitRemodeledProperty
        , @PaintDate = @New_PaintDate
        , @RemodelPropertyDate = @New_RemodelPropertyDate
        , @isExposedtoPeelingPaint = @New_isExposedtoPeelingPaint
        , @isTakingVitamins = @New_isTakingVitamins
        , @NursingMother = @New_NursingMother
        , @NursingInfant = @New_NursingInfant
        , @Pregnant = @New_Pregnant
        , @isUsingPacifier = @New_isUsingPacifier
        , @isUsingBottle = @New_isUsingBottle
        , @BitesNails = @New_BitesNails
        , @NonFoodEating = @New_NonFoodEating
        , @NonFoodinMouth = @New_NonFoodinMouth
        , @EatOutside = @New_EatOutside
        , @Suckling = @New_Suckling
        , @Mouthing = @New_Mouthing
        , @FrequentHandWashing = @New_FrequentHandWashing
        , @VisitsOldHomes = @New_VisitsOldHomes
        , @DaycareID = @New_DaycareID
        , @QuestionnaireID = @QuestionnaireID

    IF (@New_Notes IS NOT NULL)
    EXEC      [dbo].[usp_InsertQuestionnaireNotes]
                @Questionnaire_ID = @QuestionnaireID,
                @Notes = @New_Notes,
                @InsertedNotesID = @NotesID OUTPUT

END TRY
BEGIN CATCH
    -- Call procedure to print error information.

    EXECUTE dbo.uspPrintError;

    -- Roll back any active or uncommittable transactions before
    -- inserting information in the ErrorLog.

```

Author: liam

```
IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

EXECUTE dbo.usp.LogError @ErrorLogID = @ErrorLogID OUTPUT;
THROW
END CATCH;
END

GO
```

Uses

[dbo].[Questionnaire]
[dbo].[usp_InsertQuestionnaireNotes]
[dbo].[usp.LogError]
[dbo].[uspPrintError]

Used By

[dbo].[usp_upQuestionnaireWebScreen]

[dbo].[usp_upQuestionnaireWebScreen]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@QuestionnaireID	int	4
@QuestionnaireDate	date	3
@QuestionnaireDataSourceID	int	4
@VisitRemodeledProperty	bit	1
@PaintDate	date	3
@RemodelPropertyDate	date	3
@isExposedtoPeelingPaint	bit	1
@isTakingVitamins	bit	1
@NursingMother	bit	1
@NursingInfant	bit	1
@Pregnant	bit	1
@isUsingPacifier	bit	1
@isUsingBottle	bit	1
@BitesNails	bit	1
@NonFoodEating	bit	1
@NonFoodinMouth	bit	1
@EatOutside	bit	1
@Suckling	bit	1
@Mouthing	bit	1
@FrequentHandWashing	bit	1
@VisitsOldHomes	bit	1
@DaycareID	int	4
@Notes	varchar(3000)	3000
@DEBUG	bit	1

SQL Script

Author: liam

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150618
-- Description: stored procedure to update
--          questionnaire data
-- =====

CREATE PROCEDURE [dbo].[usp_upQuestionnaireWebScreen]
    -- Add the parameters for the stored procedure here

    @QuestionnaireID int = NULL,
    @QuestionnaireDate date = NULL,
    @QuestionnaireDataSourceID int = NULL,
    @VisitRemodeledProperty bit = NULL,
    @PaintDate date = NULL,
    @RemodelPropertyDate date = NULL,
    @isExposedtoPeelingPaint bit = NULL,
    @isTakingVitamins bit = NULL,
    @NursingMother bit = NULL,
    @NursingInfant bit = NULL,
    @Pregnant bit = NULL,
    @isUsingPacifier bit = NULL,
    @isUsingBottle bit = NULL,
    @BitesNails bit = NULL,
    @NonFoodEating bit = NULL,
    @NonFoodinMouth bit = Null,
    @EatOutside bit = NULL,
    @Suckling bit = NULL,
    @Mouthing bit = NULL,
    @FrequentHandWashing bit = NULL,
    @VisitsOldHomes bit = NULL,
    @DaycareID int = NULL,
    @Notes varchar(3000) = NULL,
    @DEBUG BIT = 0

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    SET NOCOUNT ON;

    BEGIN
        DECLARE @ErrorLogID int, @updateQuestionnaireReturnValue int;

        -- If no family ID was passed in exit
    END

```

```
IF (@QuestionnaireID IS NULL)
BEGIN
    RAISERROR ('Questionnaire ID must be supplied', 11, -1);
    RETURN;
END;

BEGIN TRY
    -- update questionnaire

    EXEC    @updateQuestionnaireReturnValue = [dbo].[usp_upQuestionnaire]
                @QuestionnaireID =
@QuestionnaireID,
                @New_QuestionnaireDate =
@QuestionnaireDate,
                @New_QuestionnaireDataSourceID =
@QuestionnaireDataSourceID,
                @New_VisitRemodeledProperty =
@VisitRemodeledProperty,
                @New_PaintDate = @PaintDate,
                @New_RemodelPropertyDate =
@RemodelPropertyDate,
                @New_isExposedtoPeelingPaint =
@isExposedtoPeelingPaint,
                @New_isTakingVitamins = @is-
TakingVitamins,
                @New_NursingMother = @Nursing-
Mother,
                @New_NursingInfant = @Nursing-
Infant,
                @New_Pregnant = @Pregnant,
                @New_isUsingPacifier = @isUsing-
Pacifier,
                @New_isUsingBottle = @isUsing-
Bottle,
                @New_BitesNails = @BitesNails,
                @New_NonFoodEating = @NonFood-
Eating,
                @New_NonFoodinMouth = @Non-
FoodinMouth,
                @New_EatOutside = @EatOutside,
                @New_Suckling = @Suckling,
                @New_Mouthing = @Mouthing,
                @New_FrequentHandWashing =
@FrequentHandWashing,
                @New_VisitsOldHomes = @Visits-
OldHomes,
                @New_DaycareID = @DaycareID,
                @New_Notes = @Notes,
                @DEBUG = @DEBUG

    END TRY
    BEGIN CATCH -- insert person

        -- Call procedure to print error information.

        EXECUTE dbo.uspPrintError;
```

```
-- Roll back any active or uncommittable transactions before
-- inserting information in the ErrorLog.

IF XACT_STATE() <> 0
BEGIN
    ROLLBACK TRANSACTION;
END

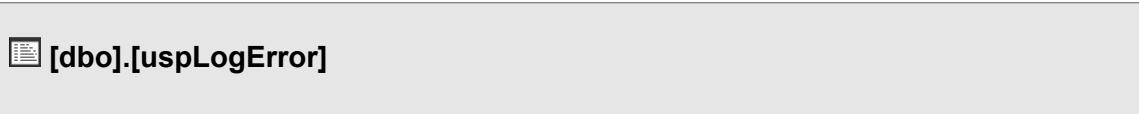
EXECUTE dbo.uspLogError @ErrorLogID = @ErrorLogID OUTPUT;
RETURN ERROR_NUMBER()
END CATCH; -- insert new person

END
END

GO
```

Uses

[dbo].[usp_upQuestionnaire]
[dbo].[uspLogError]
[dbo].[uspPrintError]



Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)	Direction
@ErrorLogID	int	4	Out

SQL Script

```

CREATE PROCEDURE [dbo].[usp.LogError]
    @ErrorLogID [int] = 0 OUTPUT -- Contains the ErrorLogID of the row inserted
                                -- by usp.LogError in the ErrorLog table.

AS
BEGIN
    SET NOCOUNT ON;

    -- Output parameter value of 0 indicates that error
    -- information was not logged.

    SET @ErrorLogID = 0;

    BEGIN TRY
        -- Return if there is no error information to log.

        IF ERROR_NUMBER() IS NULL
            RETURN;

        -- Return if inside an uncommittable transaction.

        -- Data insertion/modification is not allowed when
        -- a transaction is in an uncommittable state.

        IF XACT_STATE() = -1
        BEGIN
            PRINT 'Cannot log error since the current transaction is in an

```

```

uncommittable state. '
        + 'Rollback the transaction before executing uspLogError in order to
successfully log error information.';

    RETURN;
END;

INSERT [dbo].[ErrorLog]
(
    [UserName],
    [ErrorNumber],
    [ErrorSeverity],
    [ErrorState],
    [ErrorProcedure],
    [ErrorLine],
    [ErrorMessage]
)
VALUES
(
    CONVERT(sysname, CURRENT_USER),
    ERROR_NUMBER(),
    ERROR_SEVERITY(),
    ERROR_STATE(),
    ERROR_PROCEDURE(),
    ERROR_LINE(),
    ERROR_MESSAGE()
);

-- Pass back the ErrorLogID of the row inserted

SELECT @ErrorLogID = @@IDENTITY;

END TRY
BEGIN CATCH
    PRINT 'An error occurred in stored procedure uspLogError: ';
    EXECUTE [dbo].[uspPrintError];
    RETURN -1;
END CATCH
END;
GO

```

Uses

[dbo].[ErrorLog]
[dbo].[uspPrintError]

Used By

[dbo].[DELETE_usp_InsertPersonStatus]
[dbo].[DELETE_usp_SICountClients]
[dbo].[usp_InsertAccessAgreement]
[dbo].[usp_InsertAccessPurpose]
[dbo].[usp_InsertArea]
[dbo].[usp_InsertBloodTestResults]

```
[dbo].[usp_InsertBloodTestResultsNotes]
[dbo].[usp_InsertCleanupStatus]
[dbo].[usp_InsertConstructionType]
[dbo].[usp_InsertContractor]
[dbo].[usp_InsertContractortoProperty]
[dbo].[usp_InsertContractortoRemediation]
[dbo].[usp_InsertContractortoRemediationActionPlan]
[dbo].[usp_InsertCountry]
[dbo].[usp_InsertDaycare]
[dbo].[usp_InsertDaycarePrimaryContact]
[dbo].[usp_InsertDaycaretoProperty]
[dbo].[usp_InsertEmployer]
[dbo].[usp_InsertEmployertoProperty]
[dbo].[usp_InsertEnvironmentalInvestigation]
[dbo].[usp_InsertEthnicity]
[dbo].[usp_InsertFamily]
[dbo].[usp_InsertFamilyNotes]
[dbo].[usp_InsertFamilytoPhoneNumber]
[dbo].[usp_InsertFamilytoProperty]
[dbo].[usp_InsertForeignFood]
[dbo].[usp_InsertForeignFoodtoCountry]
[dbo].[usp_InsertGiftCard]
[dbo].[usp_InsertHobby]
[dbo].[usp_InsertHomeRemedies]
[dbo].[usp_InsertHouseholdSourcesofLead]
[dbo].[usp_InsertInsuranceProvider]
[dbo].[usp_InsertLab]
[dbo].[usp_InsertLabNotes]
[dbo].[usp_InsertLanguage]
[dbo].[usp_InsertMedium]
[dbo].[usp_InsertMediumSampleResults]
[dbo].[usp_InsertMediumSampleResultsNotes]
[dbo].[usp_InsertNewBloodLeadTestResultsWebScreen]
[dbo].[usp_InsertNewClientWebScreen]
[dbo].[usp_InsertNewFamilyWebScreen]
[dbo].[usp_InsertNewQuestionnaireWebScreen]
[dbo].[usp_InsertOccupation]
[dbo].[usp_InsertPerson]
[dbo].[usp_InsertPersonHobbyNotes]
[dbo].[usp_InsertPersonNotes]
[dbo].[usp_InsertPersonReleaseNotes]
[dbo].[usp_InsertPersonstoAccessAgreement]
[dbo].[usp_InsertPersonstoDaycare]
[dbo].[usp_InsertPersonstoEmployer]
[dbo].[usp_InsertPersonstoEthnicity]
[dbo].[usp_InsertPersonstoFamily]
[dbo].[usp_InsertPersonstoForeignFood]
[dbo].[usp_InsertPersonstoHobby]
[dbo].[usp_InsertPersonstoHomeRemedy]
[dbo].[usp_InsertPersonstoInsurance]
[dbo].[usp_InsertPersonstoLanguage]
[dbo].[usp_InsertPersonstoOccupation]
[dbo].[usp_InsertPersonstoPerson]
```

[dbo].[usp_InsertPersonToPhoneNumber]
[dbo].[usp_InsertPersonToProperty]
[dbo].[usp_InsertPersonToTravelCountry]
[dbo].[usp_InsertPersonTravelNotes]
[dbo].[usp_InsertPhoneNumber]
[dbo].[usp_InsertPhoneNumberType]
[dbo].[usp_InsertProperty]
[dbo].[usp_InsertPropertyNotes]
[dbo].[usp_InsertPropertySampleResults]
[dbo].[usp_InsertPropertySampleResultsNotes]
[dbo].[usp_InsertPropertytoCleanupStatus]
[dbo].[usp_InsertPropertytoHouseholdSourcesofLead]
[dbo].[usp_InsertPropertytoMedium]
[dbo].[usp_InsertQuestionnaire]
[dbo].[usp_InsertQuestionnaireNotes]
[dbo].[usp_InsertRemediation]
[dbo].[usp_InsertRemediationActionPlan]
[dbo].[usp_InsertRemediationNotes]
[dbo].[usp_InsertSampleLevelCategory]
[dbo].[usp_InsertSampleType]
[dbo].[usp_InsertStatus]
[dbo].[usp_InsertTravelNotes]
[dbo].[usp_SLAIBloodTestResults]
[dbo].[usp_SLAIBloodTestResults2]
[dbo].[usp_SIClientFollowUp]
[dbo].[usp_SICountAdults]
[dbo].[usp_SICountBloodLeadLevels]
[dbo].[usp_SICountBloodTests]
[dbo].[usp_SICountClients]
[dbo].[usp_SICountFamilyMembers]
[dbo].[usp_SICountHomeVisitSoilSample]
[dbo].[usp_SICountNewClients]
[dbo].[usp_SICountNewPeople]
[dbo].[usp_SICountNursingInfants]
[dbo].[usp_SICountNursingMothers]
[dbo].[usp_SICountPeople]
[dbo].[usp_SICountPeopleByLastName]
[dbo].[usp_SICountPregnantWomen]
[dbo].[usp_SIEditBloodTestResultsWebScreenInformation]
[dbo].[usp_SIEditClientInfoWebScreenInformation]
[dbo].[usp_SIEditFamilyWebScreenInformation]
[dbo].[usp_SIEditPropertyWebScreenInformation]
[dbo].[usp_SIEditQuestionnaireWebScreenInformation]
[dbo].[usp_SIFamilyMembers]
[dbo].[usp_SIFamilyNameToProperty]
[dbo].[usp_SLInsertedData]
[dbo].[usp_SLInsertedDataSimplified]
[dbo].[usp_SLLListAllFamilyMembers]
[dbo].[usp_SIListClientsByCreatedDate]
[dbo].[usp_SIListClientsByModifiedDate]
[dbo].[usp_SIListFamilies]
[dbo].[usp_SIListFamilyMembers]
[dbo].[usp_SIListNursingWomenByCreateDateRange]

```
[dbo].[usp_SIListPeoplebyCreateDateRange]
[dbo].[usp_SLListPotentialDuplicatePeople]
[dbo].[usp_SLListPotentialDuplicateProperties]
[dbo].[usp_SIListPregnantWomenbyCreateDateRange]
[dbo].[usp_SLMostRecentBloodTestResults]
[dbo].[usp_SIPersonNotes]
[dbo].[usp_SIPersontoEthnicity]
[dbo].[usp_SIPersontoLanguage]
[dbo].[usp_SIRelationshipTypes]
[dbo].[usp_SISummaryReport]
[dbo].[usp_SISummaryReport_MetaData]
[dbo].[usp_upBloodTestResults]
[dbo].[usp_upBloodTestResultsWebScreen]
[dbo].[usp_upClientFlag]
[dbo].[usp_upClientWebScreen]
[dbo].[usp_upFamily]
[dbo].[usp_upFamilytoProperty]
[dbo].[usp_upFamilyWebScreen]
[dbo].[usp_upOccupation]
[dbo].[usp_upPerson]
[dbo].[usp_upProperty]
[dbo].[usp_upQuestionnaire]
[dbo].[usp_upQuestionnaireWebScreen]
```

[dbo].[uspPrintError]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

SQL Script

```

CREATE PROCEDURE [dbo] . [uspPrintError]
AS
BEGIN
    SET NOCOUNT ON;

    -- Print error information.

    PRINT 'Error ' + CONVERT(varchar(50), ERROR_NUMBER()) +
        ', Severity ' + CONVERT(varchar(5), ERROR_SEVERITY()) +
        ', State ' + CONVERT(varchar(5), ERROR_STATE()) +
        ', Procedure ' + ISNULL(ERROR_PROCEDURE(), '-') +
        ', Line ' + CONVERT(varchar(5), ERROR_LINE());
    PRINT ERROR_MESSAGE();
END;
GO

```

Used By

[\[dbo\].\[DELETE_usp_InsertPersonToStatus\]](#)
[\[dbo\].\[DELETE_usp_SICountClients\]](#)
[\[dbo\].\[usp_InsertAccessAgreement\]](#)
[\[dbo\].\[usp_InsertAccessPurpose\]](#)
[\[dbo\].\[usp_InsertArea\]](#)
[\[dbo\].\[usp_InsertBloodTestResults\]](#)
[\[dbo\].\[usp_InsertBloodTestResultsNotes\]](#)
[\[dbo\].\[usp_InsertCleanupStatus\]](#)
[\[dbo\].\[usp_InsertConstructionType\]](#)
[\[dbo\].\[usp_InsertContractor\]](#)
[\[dbo\].\[usp_InsertContractortoProperty\]](#)
[\[dbo\].\[usp_InsertContractortoRemediation\]](#)
[\[dbo\].\[usp_InsertContractortoRemediationActionPlan\]](#)
[\[dbo\].\[usp_InsertCountry\]](#)
[\[dbo\].\[usp_InsertDaycare\]](#)
[\[dbo\].\[usp_InsertDaycarePrimaryContact\]](#)
[\[dbo\].\[usp_InsertDaycaretoProperty\]](#)
[\[dbo\].\[usp_InsertEmployer\]](#)

Author: liam

```
[dbo].[usp_InsertEmployertoProperty]
[dbo].[usp_InsertEnvironmentalInvestigation]
[dbo].[usp_InsertEthnicity]
[dbo].[usp_InsertFamily]
[dbo].[usp_InsertFamilyNotes]
[dbo].[usp_InsertFamilytoPhoneNumber]
[dbo].[usp_InsertFamilytoProperty]
[dbo].[usp_InsertForeignFood]
[dbo].[usp_InsertForeignFoodtoCountry]
[dbo].[usp_InsertGiftCard]
[dbo].[usp_InsertHobby]
[dbo].[usp_InsertHomeRemedies]
[dbo].[usp_InsertHouseholdSourcesofLead]
[dbo].[usp_InsertInsuranceProvider]
[dbo].[usp_InsertLab]
[dbo].[usp_InsertLabNotes]
[dbo].[usp_InsertLanguage]
[dbo].[usp_InsertMedium]
[dbo].[usp_InsertMediumSampleResults]
[dbo].[usp_InsertMediumSampleResultsNotes]
[dbo].[usp_InsertNewBloodLeadTestResultsWebScreen]
[dbo].[usp_InsertNewClientWebScreen]
[dbo].[usp_InsertNewFamilyWebScreen]
[dbo].[usp_InsertNewQuestionnaireWebScreen]
[dbo].[usp_InsertOccupation]
[dbo].[usp_InsertPerson]
[dbo].[usp_InsertPersonHobbyNotes]
[dbo].[usp_InsertPersonNotes]
[dbo].[usp_InsertPersonReleaseNotes]
[dbo].[usp_InsertPersonstoAccessAgreement]
[dbo].[usp_InsertPersonstoDaycare]
[dbo].[usp_InsertPersonstoEmployer]
[dbo].[usp_InsertPersonstoEthnicity]
[dbo].[usp_InsertPersonstoFamily]
[dbo].[usp_InsertPersonstoForeignFood]
[dbo].[usp_InsertPersonstoHobby]
[dbo].[usp_InsertPersonstoHomeRemedy]
[dbo].[usp_InsertPersonstoInsurance]
[dbo].[usp_InsertPersonstoLanguage]
[dbo].[usp_InsertPersonstoOccupation]
[dbo].[usp_InsertPersonstoPerson]
[dbo].[usp_InsertPersonstoPhoneNumber]
[dbo].[usp_InsertPersonstoProperty]
[dbo].[usp_InsertPersonstoTravelCountry]
[dbo].[usp_InsertPersonTravelNotes]
[dbo].[usp_InsertPhoneNumber]
[dbo].[usp_InsertPhoneNumberType]
[dbo].[usp_InsertProperty]
[dbo].[usp_InsertPropertyNotes]
[dbo].[usp_InsertPropertySampleResults]
[dbo].[usp_InsertPropertySampleResultsNotes]
[dbo].[usp_InsertPropertytoCleanupStatus]
[dbo].[usp_InsertPropertytoHouseholdSourcesofLead]
```

```
[dbo].[usp_InsertPropertytoMedium]
[dbo].[usp_InsertQuestionnaire]
[dbo].[usp_InsertQuestionnaireNotes]
[dbo].[usp_InsertRemediation]
[dbo].[usp_InsertRemediationActionPlan]
[dbo].[usp_InsertRemediationNotes]
[dbo].[usp_InsertSampleLevelCategory]
[dbo].[usp_InsertSampleType]
[dbo].[usp_InsertStatus]
[dbo].[usp_InsertTravelNotes]
[dbo].[usp_SLAllBloodTestResults]
[dbo].[usp_SLAllBloodTestResults2]
[dbo].[usp_SIClientFollowUp]
[dbo].[usp_SICountAdults]
[dbo].[usp_SICountBloodLeadLevels]
[dbo].[usp_SICountBloodTests]
[dbo].[usp_SICountClients]
[dbo].[usp_SICountFamilyMembers]
[dbo].[usp_SICountHomeVisitSoilSample]
[dbo].[usp_SICountNewClients]
[dbo].[usp_SICountNewPeople]
[dbo].[usp_SICountNursingInfants]
[dbo].[usp_SICountNursingMothers]
[dbo].[usp_SICountPeople]
[dbo].[usp_SICountPeopleByLastName]
[dbo].[usp_SICountPregnantWomen]
[dbo].[usp_SIEditBloodTestResultsWebScreenInformation]
[dbo].[usp_SIEditClientInfoWebScreenInformation]
[dbo].[usp_SIEditFamilyWebScreenInformation]
[dbo].[usp_SIEditPropertyWebScreenInformation]
[dbo].[usp_SIEditQuestionnaireWebScreenInformation]
[dbo].[usp_SIFamilyMembers]
[dbo].[usp_SIFamilyNameToProperty]
[dbo].[usp_SLInsertedData]
[dbo].[usp_SLInsertedDataSimplified]
[dbo].[usp_SLLListAllFamilyMembers]
[dbo].[usp_SIListClientsByCreatedate]
[dbo].[usp_SIListClientsByModifieddate]
[dbo].[usp_SIListFamilies]
[dbo].[usp_SIListFamilyMembers]
[dbo].[usp_SIListNursingWomenbyCreateDateRange]
[dbo].[usp_SIListPeoplebyCreateDateRange]
[dbo].[usp_SLLListPotentialDuplicatePeople]
[dbo].[usp_SLLListPotentialDuplicateProperties]
[dbo].[usp_SIListPregnantWomenbyCreateDateRange]
[dbo].[usp_SLMostRecentBloodTestResults]
[dbo].[usp_SIPersonNotes]
[dbo].[usp_SIPersonToEthnicity]
[dbo].[usp_SIPersonToLanguage]
[dbo].[usp_SIRelationShipTypes]
[dbo].[usp_SISummaryReport]
[dbo].[usp_SISummaryReport_MetaData]
[dbo].[usp_upBloodTestResults]
```

```
[dbo].[usp_upBloodTestResultsWebScreen]  
[dbo].[usp_upClientFlag]  
[dbo].[usp_upClientWebScreen]  
[dbo].[usp_upFamily]  
[dbo].[usp_upFamilytoProperty]  
[dbo].[usp_upFamilyWebScreen]  
[dbo].[usp_upOccupation]  
[dbo].[usp_upPerson]  
[dbo].[usp_upProperty]  
[dbo].[usp_upQuestionnaire]  
[dbo].[usp_upQuestionnaireWebScreen]  
[dbo].[uspLogError]
```

Scalar-valued Functions

Objects

Name
dbo.RemoveSpecialChars
dbo.udf_CalculateAge
dbo.udf_DateInThePast
dbo.udf_DoesPropertyExist
dbo.udf_SIFamilyPhoneNumber

[dbo].[RemoveSpecialChars]

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True
Schema Bound	True

Parameters

Name	Data Type	Max Length (Bytes)
@s	varchar(256)	256

SQL Script

```
-- Removes special characters from a string value.

-- All characters except 0-9, a-z and A-Z are removed and

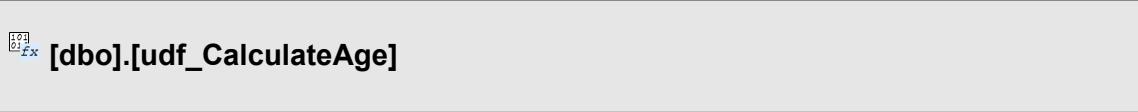
-- the remaining characters are returned.

-- Author: Christian d'Heureuse, www.source-code.biz

CREATE function [dbo].[RemoveSpecialChars] (@s varchar(256)) returns varchar(256)
    with schemabinding
begin
    if @s is null
        return null
    declare @s2 varchar(256)
    set @s2 = ''
    declare @l int
    set @l = len(@s)
    declare @p int
    set @p = 1
    while @p <= @l begin
        declare @c int
        set @c = ascii(substring(@s, @p, 1))
        if @c between 48 and 57 or @c between 65 and 90 or @c between 97 and 122
            set @s2 = @s2 + char(@c)
        set @p = @p + 1
    end
    if len(@s2) = 0
        return null
    return @s2
```

Project> (local)> User databases> LCCHPDev> Programmability> Functions> Scalar-valued Functions>
dbo.RemoveSpecialChars

```
end  
GO
```



Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@BirthDate	datetime	8
@CurrentDate	datetime	8

SQL Script

```
CREATE FUNCTION [dbo].[udf_CalculateAge]
(
    @BirthDate datetime = NULL,
    @CurrentDate datetime = NULL
)
RETURNS int

AS

BEGIN

    IF @BirthDate IS NULL
        RETURN -1;

    IF @CurrentDate IS NULL
        SET @CurrentDate = GetDate();

    IF @BirthDate > @CurrentDate
        RETURN 0

    DECLARE @Age int
    SELECT @Age = DATEDIFF(YY, @BirthDate, @CurrentDate) -
        CASE WHEN (
            (MONTH(@BirthDate)*100 + DAY(@BirthDate)) >
            (MONTH(@CurrentDate)*100 + DAY(@CurrentDate))
        ) THEN 1 ELSE 0 END
    RETURN @Age
```

```
END
```

```
GO
```

Used By

[dbo].[Person]
[dbo].[Questionnaire]
[dbo].[usp_SICountAdults]
[dbo].[usp_SISummaryReport]

[dbo].[udf_DateInThePast]	

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@CheckDate	date	3

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150220
-- Description:    function to ensure date is less than current date
-- =====

CREATE FUNCTION [dbo].[udf_DateInThePast]
(
    -- Add the parameters for the function here
    @CheckDate date
)
RETURNS bit
AS
BEGIN
    -- Declare the return variable here
    DECLARE @Result bit

    -- Add the T-SQL statements to compute the return value here

    IF (@CheckDate < GetDate())
        SET @Result = 1;
    ELSE
        SET @Result = 0;

    -- Return the result of the function

```

Author: liam

```
    RETURN @Result  
  
END  
GO
```

Used By

[dbo].[BloodTestResults]
[dbo].[MediumSampleResults]
[dbo].[Person]
[dbo].[PropertySampleResults]
[dbo].[Questionnaire]
[dbo].[Remediation]
[dbo].[RemediationActionPlan]

[dbo].[udf_DoesPropertyExist]	
-------------------------------	--

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@AddressLine1	varchar(100)	100
@AddressLine2	varchar(100)	100
@City	varchar(50)	50
@State	char(2)	2
@ZipCode	varchar(12)	12

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150322
-- Description: Function to check for duplicate property
-- =====

CREATE FUNCTION [dbo].[udf_DoesPropertyExist]
(
    -- Add the parameters for the function here

    @AddressLine1 varchar(100),
    @AddressLine2 varchar(100) = NULL,
    @City varchar(50),
    @State char(2),
    @ZipCode varchar(12)
)
RETURNS int
AS
BEGIN
    -- Declare the return variable here

```

Author: liam

```
DECLARE @PropertyID int

IF (@AddressLine2 IS NULL)
    SELECT @PropertyID = PropertyID from Property where
        -- (dbo.RemoveSpecialChars(AddressLine1) = dbo.RemoveSpecialChars(@Address-
Line1))

        AddressLine1 = @AddressLine1
        AND (AddressLine2 = '')
        AND (City = @City )
        and ([State] = @State and Zipcode = @ZipCode)
ELSE
    SELECT @PropertyID = PropertyID from Property where
        --(dbo.RemoveSpecialChars(AddressLine1) = dbo.RemoveSpecialChars(@Address-
Line1))

        AddressLine1 = @AddressLine1
        --AND (dbo.RemoveSpecialChars(AddressLine2) = dbo.RemoveSpecial-
        Chars (@AddressLine2))

        AND AddressLine2 = @AddressLine2
        AND (City = @City )
        and ([State] = @State and Zipcode = @ZipCode)

-- Return the result of the function

RETURN @PropertyID

END
GO
```

Uses

[dbo].[Property]

Used By

[dbo].[usp_InsertNewFamilyWebScreen]
[dbo].[usp_InsertProperty]

[dbo].[udf_SIFamilyPhoneNumber]	
---------------------------------	--

Properties

Property	Value
ANSI Nulls On	True
Quoted Identifier On	True

Parameters

Name	Data Type	Max Length (Bytes)
@Family_ID	int	4
@PhoneNumberTypeID	tinyint	1

SQL Script

```
-- =====
-- Author:      Liam Thier
-- Create date: 20150405
-- Description: select specific phone number of
--               specified type for specific family
-- =====

CREATE FUNCTION [dbo].[udf_SIFamilyPhoneNumber]
(
    -- Add the parameters for the function here
    @Family_ID int,
    @PhoneNumberTypeID tinyint
)
RETURNS bigint
AS
BEGIN
    -- Declare the return variable here

    DECLARE @PhoneNumber bigint--, @PhoneNumberTypeID tinyint; --, @Family_ID int,
    @PhoneNumberTypeID tinyint;

    --SET @PhoneNumberTypeID = 1
```

```
-- Add the T-SQL statements to compute the return value here

--     SELECT @PhoneNumber = @Family_ID

Select @PhoneNumber = [P].[PhoneNumber] from [PhoneNumber] AS [P]
JOIN [FamilytoPhoneNumber] AS [P2N] ON [P].[PhoneNumberID] = [P2N].[PhoneNumberID]
JOIN [Family] AS [F] ON [P2N].[FamilyID] = [F].[FamilyID]
WHERE [F].[FamilyID] = @Family_ID and [P2N].[NumberPriority] = @PhoneNumberTypeID

-- Select @PhoneNumber

-- Return the result of the function

RETURN @PhoneNumber

END
GO
```

Uses

[dbo].[Family]
[dbo].[FamilytoPhoneNumber]
[dbo].[PhoneNumber]

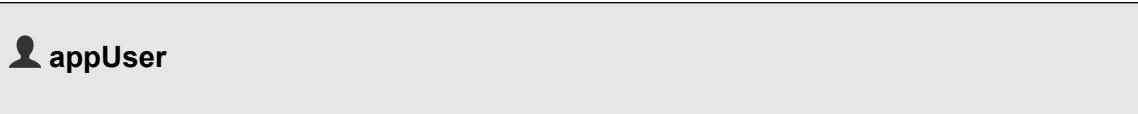
Used By

[dbo].[usp_SIEditFamilyWebScreenInformation]

Users

Objects

Name
appUser
WIN-1M8NQQ69OEH\SQLMaintenance



Properties

Property	Value
Type	SqlUser
Login Name	appUser
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
IF NOT EXISTS (SELECT * FROM master.dbo.syslogins WHERE loginname = N'appUser')
CREATE LOGIN [appUser] WITH PASSWORD = 'p@ssw0rd'
GO
CREATE USER [appUser] FOR LOGIN [appUser]
GO
```

WIN-1M8NQQ69OEH\SQLMaintenenace

Properties

Property	Value
Type	WindowsUser
Login Name	WIN-1M8NQQ69OEH\SQLMaintenenace
Default Schema	dbo

Database Level Permissions

Type	Action
CONNECT	Grant

SQL Script

```
IF NOT EXISTS (SELECT * FROM master.dbo.syslogins WHERE loginname = N'WIN-1M8NQQ69OEH\SQLMaintenenace')
CREATE LOGIN [WIN-1M8NQQ69OEH\SQLMaintenenace] FROM WINDOWS
GO
CREATE USER [WIN-1M8NQQ69OEH\SQLMaintenenace] FOR LOGIN [WIN-1M8NQQ69OEH\SQLMaintenenace]
GO
```

Database Roles

Objects

Name
db_accessadmin
db_backupoperator
db_datareader
db_datawriter
db_ddladmin
db_denydatareader
db_denydatawriter
db_owner
db_securityadmin
public

db_accessadmin

Properties

Property	Value
Owner	dbo

db_backupoperator

Properties

Property	Value
Owner	dbo

Members

- WIN-1M8NQQ69OEH\SQLMaintenenace

SQL Script

```
EXEC sp_addrolemember N'db_backupoperator', N'WIN-1M8NQQ69OEH\SQLMaintenenace'  
GO
```

Uses

WIN-1M8NQQ69OEH\SQLMaintenenace

db_datareader

Properties

Property	Value
Owner	dbo

Members

- appUser

SQL Script

```
EXEC sp_addrolemember N'db_datareader', N'appUser'  
GO
```

Uses

appUser

db_datawriter

Author: liam

Properties

Property	Value
Owner	dbo

Members

- appUser

SQL Script

```
EXEC sp_addrolemember N'db_datawriter', N'appUser'  
GO
```

Uses

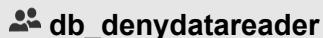
appUser



db_ddladmin

Properties

Property	Value
Owner	dbo



db_denydatareader

Properties

Property	Value
Owner	dbo

db_denydatawriter

Properties

Property	Value
Owner	dbo

db_owner

Properties

Property	Value
Owner	dbo

Members

- appUser

SQL Script

```
EXEC sp_addrolemember N'db_owner', N'appUser'  
GO
```

Uses

appUser

db_securityadmin

Properties

Property	Value
Owner	dbo



Properties

Property	Value
Owner	dbo