# 

[SQL Reply Tables](#_egk4xr9yhhu1)

[SQL Reply Table Triggers](#_5etv6eg6xb24)

[SQL Functions](#_seitfqiwhq87)

[fGetMediaTypes](#_w1mkssz5su00)

# SQL Reply Tables

USE Reply

go

-- \*\*\*\*\*\*\*\*\*

-- MediaType

-- \*\*\*\*\*\*\*\*\*

if exists(select name from sysobjects where name = 'MediaType')

drop table MediaType

go

CREATE TABLE MediaType

(

id tinyint IDENTITY,

mediaTypeName varchar(100) NOT NULL,

sortSeq tinyint NULL,

active bit NOT NULL DEFAULT 1,

createUser varchar(100) NOT NULL DEFAULT SYSTEM\_USER,

createDate datetime NOT NULL DEFAULT GETDATE(),

updateUser varchar(100) NOT NULL DEFAULT SYSTEM\_USER,

updateDate datetime NOT NULL DEFAULT GETDATE(),

CONSTRAINT PK\_MediaType\_id PRIMARY KEY NONCLUSTERED (id)

)

go

-- \*\*\*\*\*\*\*\*\*\*\*

-- MediaSource

-- \*\*\*\*\*\*\*\*\*\*\*

if exists(select name from sysobjects where name = 'MediaSource')

drop table MediaSource

go

CREATE TABLE MediaSource

(

id int IDENTITY,

mediaSourceName varchar(100) NOT NULL,

mediaTypeId tinyint NOT NULL,

sortSeq int NULL,

active bit NOT NULL DEFAULT 1,

createUser varchar(100) NOT NULL DEFAULT SYSTEM\_USER,

createDate datetime NOT NULL DEFAULT GETDATE(),

updateUser varchar(100) NOT NULL DEFAULT SYSTEM\_USER,

updateDate datetime NOT NULL DEFAULT GETDATE(),

CONSTRAINT PK\_MediaSource\_id PRIMARY KEY NONCLUSTERED (id),

CONSTRAINT FK\_MediaSource\_mediaTypeId FOREIGN KEY (mediaTypeId) REFERENCES MediaType(id)

)

go

-- \*\*\*\*\*\*\*\*

-- Category

-- \*\*\*\*\*\*\*\*

if exists(select name from sysobjects where name = 'Category')

drop table Category

go

CREATE TABLE Category

(

id tinyint IDENTITY,

categoryName varchar(100) NOT NULL,

sortSeq tinyint NULL,

active bit NOT NULL DEFAULT 1,

createUser varchar(100) NOT NULL DEFAULT SYSTEM\_USER,

createDate datetime NOT NULL DEFAULT GETDATE(),

updateUser varchar(100) NOT NULL DEFAULT SYSTEM\_USER,

updateDate datetime NOT NULL DEFAULT GETDATE(),

CONSTRAINT PK\_Category\_id PRIMARY KEY NONCLUSTERED (id)

)

go

-- \*\*\*\*\*\*\*\*

-- Reporter

-- \*\*\*\*\*\*\*\*

if exists(select name from sysobjects where name = 'Reporter')

drop table Reporter

go

CREATE TABLE Reporter

(

id int IDENTITY,

reporterFirstName varchar(50) NOT NULL,

reporterLastName varchar(50) NOT NULL,

sortSeq int NULL,

active bit NOT NULL DEFAULT 1,

createUser varchar(100) NOT NULL DEFAULT SYSTEM\_USER,

createDate datetime NOT NULL DEFAULT GETDATE(),

updateUser varchar(100) NOT NULL DEFAULT SYSTEM\_USER,

updateDate datetime NOT NULL DEFAULT GETDATE(),

CONSTRAINT PK\_Reporter\_id PRIMARY KEY NONCLUSTERED (id)

)

go

-- \*\*\*\*\*\*\*\*\*\*

-- Subscriber

-- \*\*\*\*\*\*\*\*\*\*

if exists(select name from sysobjects where name = 'Subscriber')

drop table Subscriber

go

CREATE TABLE Subscriber

(

id int IDENTITY,

subscriberFirstName varchar(50) NOT NULL,

subscriberLastName varchar(50) NOT NULL,

subscriberEmailAddress varchar(255) NOT NULL,

sortSeq int NULL,

active bit NOT NULL DEFAULT 1,

createUser varchar(100) NOT NULL DEFAULT SYSTEM\_USER,

createDate datetime NOT NULL DEFAULT GETDATE(),

updateUser varchar(100) NOT NULL DEFAULT SYSTEM\_USER,

updateDate datetime NOT NULL DEFAULT GETDATE(),

CONSTRAINT PK\_Subscriber\_id PRIMARY KEY NONCLUSTERED (id)

)

go

-- \*\*\*\*\*\*\*

-- Article

-- \*\*\*\*\*\*\*

if exists(select name from sysobjects where name = 'Article')

drop table Article

go

CREATE TABLE Article

(

id int IDENTITY,

mediaTypeId tinyint NOT NULL,

mediaSourceId int NULL,

otherMediaSourceName varchar(200) NOT NULL,

title varchar(500) NOT NULL,

reporterId int NULL,

otherReporterName varchar(200) NULL,

articleDate date NOT NULL,

articleURL varchar(500) NULL,

disabled bit NOT NULL DEFAULT 0,

subscriberId int NOT NULL,

createDate datetime NOT NULL DEFAULT GETDATE(),

updateUser varchar(100) NOT NULL DEFAULT SYSTEM\_USER,

updateDate datetime NOT NULL DEFAULT GETDATE(),

CONSTRAINT PK\_Article\_id PRIMARY KEY NONCLUSTERED (id),

CONSTRAINT FK\_Article\_mediaTypeId FOREIGN KEY (mediaTypeId) REFERENCES MediaType(id),

CONSTRAINT FK\_Article\_mediaSourceId FOREIGN KEY (mediaSourceId) REFERENCES MediaSource(id),

CONSTRAINT FK\_Article\_reporterId FOREIGN KEY (reporterId) REFERENCES Reporter(id),

CONSTRAINT FK\_Article\_subscriberId FOREIGN KEY (subscriberId) REFERENCES Subscriber(id)

)

go

-- \*\*\*\*\*\*\*

-- Reply

-- \*\*\*\*\*\*\*

if exists(select name from sysobjects where name = 'Reply')

drop table Reply

go

CREATE TABLE Reply

(

id int IDENTITY,

articleId int NOT NULL,

categoryId tinyint NOT NULL,

subject varchar(500) NOT NULL,

replyText varchar(4000) NOT NULL,

articleThumbsUpDown tinyint NOT NULL DEFAULT 0,

disabled bit NOT NULL DEFAULT 0,

subscriberId int NOT NULL,

createDate datetime NOT NULL DEFAULT GETDATE(),

updateUser varchar(100) NOT NULL DEFAULT SYSTEM\_USER,

updateDate datetime NOT NULL DEFAULT GETDATE(),

CONSTRAINT PK\_Reply\_id PRIMARY KEY NONCLUSTERED (id),

CONSTRAINT FK\_Reply\_articleId FOREIGN KEY (articleId) REFERENCES Article(id),

CONSTRAINT FK\_Reply\_categoryId FOREIGN KEY (categoryId) REFERENCES Category(id),

CONSTRAINT FK\_Reply\_subscriberId FOREIGN KEY (subscriberId) REFERENCES Subscriber(id)

)

go

# SQL Reply Table Triggers

USE Reply

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- \*\*\*\*\*\*\*\*\*

-- MediaType

-- \*\*\*\*\*\*\*\*\*

CREATE TRIGGER trgAfterUpdate\_MediaType

ON MediaType

AFTER UPDATE

AS

UPDATE MediaType

SET updateDate = GETDATE()

FROM Inserted i

WHERE MediaType.id = i.id

go

-- \*\*\*\*\*\*\*\*\*\*\*

-- MediaSource

-- \*\*\*\*\*\*\*\*\*\*\*

CREATE TRIGGER trgAfterUpdate\_MediaSource

ON MediaSource

AFTER UPDATE

AS

UPDATE MediaSource

SET updateDate = GETDATE()

FROM Inserted i

WHERE MediaSource.id = i.id

go

-- \*\*\*\*\*\*\*\*

-- Category

-- \*\*\*\*\*\*\*\*

CREATE TRIGGER trgAfterUpdate\_Category

ON Category

AFTER UPDATE

AS

UPDATE Category

SET updateDate = GETDATE()

FROM Inserted i

WHERE Category.id = i.id

go

-- \*\*\*\*\*\*\*\*

-- Reporter

-- \*\*\*\*\*\*\*\*

CREATE TRIGGER trgAfterUpdate\_Reporter

ON Reporter

AFTER UPDATE

AS

UPDATE Reporter

SET updateDate = GETDATE()

FROM Inserted i

WHERE Reporter.id = i.id

go

-- \*\*\*\*\*\*\*\*\*\*

-- Subscriber

-- \*\*\*\*\*\*\*\*\*\*

CREATE TRIGGER trgAfterUpdate\_Subscriber

ON Subscriber

AFTER UPDATE

AS

UPDATE Subscriber

SET updateDate = GETDATE()

FROM Inserted i

WHERE Subscriber.id = i.id

go

-- \*\*\*\*\*\*\*

-- Article

-- \*\*\*\*\*\*\*

CREATE TRIGGER trgAfterUpdate\_Article

ON Article

AFTER UPDATE

AS

UPDATE Article

SET updateDate = GETDATE()

FROM Inserted i

WHERE Article.id = i.id

go

-- \*\*\*\*\*

-- Reply

-- \*\*\*\*\*

CREATE TRIGGER trgAfterUpdate\_Reply

ON Reply

AFTER UPDATE

AS

UPDATE Reply

SET updateDate = GETDATE()

FROM Inserted i

WHERE Reply.id = i.id

Go

# SQL Functions

## fGetMediaTypes

USE Reply

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCTION: fGetMediaTypes

PURPOSE: Get media types, active only or active/inactive

May 2017

Eileen Corwin

REVISIONS:

Parameters:

1) @i\_activeOnly

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Drop \*/

IF EXISTS(SELECT \* FROM Information\_Schema.Routines Where Routine\_Name = 'fGetMediaTypes')

DROP FUNCTION fGetMediaTypes

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE FUNCTION fGetMediaTypes(@i\_activeOnly bit=1)

RETURNS Table

AS

RETURN(

SELECT \*

FROM MediaType

WHERE 1 = CASE WHEN @i\_activeOnly = 1 and active = 1 THEN 1

WHEN @i\_activeOnly = 1 and active = 0 THEN 0

ELSE 1 end

)

Go

## fGetMediaSources

USE Reply

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCTION: fGetMediaSources

PURPOSE: Get media sources, active only or active/inactive, all or by media type

May 2017

Eileen Corwin

REVISIONS:

Parameters:

1) @i\_activeOnly

2) @i\_mediaTypeId (0=all)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Drop \*/

IF EXISTS(SELECT \* FROM Information\_Schema.Routines Where Routine\_Name = 'fGetMediaSources')

DROP FUNCTION fGetMediaSources

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE FUNCTION fGetMediaSources(@i\_activeOnly bit=1, @i\_mediaTypeId tinyint=0)

RETURNS Table

AS

RETURN(

SELECT \*

FROM MediaSource

WHERE 1 = CASE WHEN @i\_activeOnly = 1 and active = 1 THEN 1

WHEN @i\_activeOnly = 1 and active = 0 THEN 0

ELSE 1

END

AND

1 = CASE WHEN @i\_mediaTypeId = 0 THEN 1

WHEN mediaTypeId = @i\_mediaTypeId THEN 1

END

)

Go

## fGetCategories

USE Reply

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCTION: fGetCategories

PURPOSE: Get categories, active only or active/inactive

May 2017

Eileen Corwin

REVISIONS:

Parameters:

1) @i\_activeOnly

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Drop \*/

IF EXISTS(SELECT \* FROM Information\_Schema.Routines Where Routine\_Name = 'fGetCategories')

DROP FUNCTION fGetCategories

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE FUNCTION fGetCategories(@i\_activeOnly bit=1)

RETURNS Table

AS

RETURN(

SELECT \*

FROM Category

WHERE 1 = CASE WHEN @i\_activeOnly = 1 and active = 1 THEN 1

WHEN @i\_activeOnly = 1 and active = 0 THEN 0

ELSE 1 end

)

Go

## fGetReporters

USE Reply

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCTION: fGetReporters

PURPOSE: Get reporters, active only or active/inactive

May 2017

Eileen Corwin

REVISIONS:

Parameters:

1) @i\_activeOnly

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Drop \*/

IF EXISTS(SELECT \* FROM Information\_Schema.Routines Where Routine\_Name = 'fGetReporters')

DROP FUNCTION fGetReporters

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE FUNCTION fGetReporters(@i\_activeOnly bit=1)

RETURNS Table

AS

RETURN(

SELECT \*

FROM Reporter

WHERE 1 = CASE WHEN @i\_activeOnly = 1 and active = 1 THEN 1

WHEN @i\_activeOnly = 1 and active = 0 THEN 0

ELSE 1

END

)

Go

## fGetSubscribers

USE Reply

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCTION: fGetSubscribers

PURPOSE: Get subscribers, active only or active/inactive

May 2017

Eileen Corwin

REVISIONS:

Parameters:

1) @i\_activeOnly

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Drop \*/

IF EXISTS(SELECT \* FROM Information\_Schema.Routines Where Routine\_Name = 'fGetSubscribers')

DROP FUNCTION fGetSubscribers

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE FUNCTION fGetSubscribers(@i\_activeOnly bit=1)

RETURNS Table

AS

RETURN(

SELECT subscriberFirstName + ' ' + subscriberLastName as subscriberName

,\*

FROM Subscriber

WHERE 1 = CASE WHEN @i\_activeOnly = 1 and active = 1 THEN 1

WHEN @i\_activeOnly = 1 and active = 0 THEN 0

ELSE 1

END

)

Go