```
if not exists(select * from sys.databases where name='Names')
    create database Names
GO

use Names
GO

--DOWN
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where
CONSTRAINT_NAME='fk_name_person_id')
    alter table names drop constraint fk_name_person_id
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where
CONSTRAINT_NAME='fk_name_locale_id')
    alter table names drop constraint fk_name_locale_id
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where
CONSTRAINT_NAME='fk_name_father_person_id')
    alter table names drop constraint fk_name_father_person_id
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where
CONSTRAINT_NAME='fk_name_mother_person_id')
    alter table names drop constraint fk_name_mother_person_id
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where
CONSTRAINT_NAME='fk_name_preferred_honorific_id')
    alter table names drop constraint fk_name_preferred_honorific_id
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where
CONSTRAINT_NAME='fk_name_given_name_particle_id')
    alter table names drop constraint fk_name_given_name_particle_id
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where
CONSTRAINT_NAME='fk_name_family_name_particle_id')
    alter table names drop constraint fk_name_family_name_particle_id
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where
CONSTRAINT_NAME='fk_name_tribe_or_clan_particle_id')
    alter table names drop constraint fk_name_tribe_or_clan_particle_id
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where
CONSTRAINT_NAME='fk_name_legal_alias_particle_id')
    alter table names drop constraint fk_name_legal_alias_particle_id
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where
CONSTRAINT_NAME='fk_name_use_name_particle_id')
    alter table names drop constraint fk_name_use_name_particle_id

if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where
CONSTRAINT_NAME='fk_particles_particle_locale_id')
    alter table particles drop constraint fk_particles_particle_locale_id

if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where
CONSTRAINT_NAME='fk_particle_orders_particle_order_name_id')
    alter table particle_orders drop constraint
fk_particle_orders_particle_order_name_id
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where
CONSTRAINT_NAME='fk_particle_orders_particle_order_particle_id')
    alter table particle_orders drop constraint
fk_particle_orders_particle_order_particle_id
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS where
CONSTRAINT_NAME='fk_particle_orders_particle_order_locale_id')
    alter table particle_orders drop constraint
fk_particle_orders_particle_order_locale_id
drop function if exists get_particle_type_id
drop table if exists particle_types
drop table if exists particle_orders
drop function if exists get_particle_id_by_latin1
```

```
drop function if exists get_particle_id
drop table if exists particles
drop table if exists honorifics
drop function if exists get_locale_id
drop table if exists locales
drop table if exists names
drop table if exists persons
drop function if exists get_particle_type_id
drop function if exists get_particle_id
drop function if exists get_particle_id_by_latin1
drop function if exists get_particle_id_by_ipa
drop function if exists get_person_id
drop procedure if exists insert_piggy

GO

--UP Metadata
create table persons (
    person_id int IDENTITY not null,
    person_email varchar(50),
    constraint pk_persons_person_id primary key(person_id)
)

create table names (
    name_id int IDENTITY not null,
    name_locale_id int not null,
    name_is_legal_name bit not null,
    name_is_dead_name bit not null,
    name_preferred_locale_id int,
    name_gender_identity varchar(10),
    name_preferred_honorific_id int,
    name_preferred_pronoun_nominative nvarchar(50),
    name_preferred_pronoun_accusative nvarchar(50),
    name_preferred_pronoun_genative nvarchar(50),
    name_override_full_name nvarchar(max),
    name_override_full_name_latin1 varchar(max),
    name_override_full_name_ipa nvarchar(max),
    name_given_name_particle_id int not null,
    name_family_name_particle_id int,
    name_tribe_or_clan_particle_id int,
    name_legal_alias_particle_id int,
    name_use_name_particle_id int,
    name_person_id int not null,
    name_mother_person_id int,
    name_father_person_id int,
    constraint pk_names_name_id primary key (name_id)
)

create table locales (
    locale_id int IDENTITY not null,
    locale_language varchar(4) not null,
    locale_country varchar(4) not null,
    constraint pk_locale_locale_id primary key (locale_id),
    constraint u_locale_language_country unique (locale_language, locale_country)
)

create table honorifics (
    honorific_id int IDENTITY not null,
    honorific_text nvarchar(50) not null,
```

```
    honorific_latin1 varchar(50),
    honorific_ipa nvarchar(50),
    honorific_is_prefix bit not null,
    honorific_locale_id int not null,
    constraint pk_honorifics_honorific_id primary key (honorific_id),
    constraint u_honorifics_honorific_text unique (honorific_text),
    constraint u_honorifics_honorific_locale_id unique (honorific_locale_id)
)

create table particles (
    particle_id int IDENTITY not null,
    particle_type_id int not null,
    particle_unicode nvarchar(50) not null,
    particle_latin1 varchar(50),
    particle_ipa nvarchar(50),
    particle_locale_id int not null,
    constraint pk_particles_particle_id primary key (particle_id)
)

create table particle_orders (
    particle_order_id int IDENTITY not null,
    particle_order_name_id int not null,
    particle_order_particle_id int not null,
    particle_order_order int not null,
    particle_order_locale_id int not null,
    constraint pk_particle_orders_particle_order_id primary key (particle_order_id)
)

create table particle_types (
    particle_type_id int IDENTITY not null,
    particle_type_type varchar(50),
    constraint pk_particle_types_particle_type_id primary key (particle_type_id)
)

--UP Data
alter table NAMES
add constraint fk_name_person_id foreign key (name_person_id) references
persons(person_id)
alter table NAMES
add constraint fk_name_locale_id foreign key (name_locale_id) references
locales(locale_id)
alter table NAMES
add constraint fk_name_mother_person_id foreign key (name_mother_person_id)
references persons(person_id)
alter table NAMES
add constraint fk_name_father_person_id foreign key (name_father_person_id)
references persons(person_id)
alter table NAMES
add constraint fk_name_preferred_honorific_id foreign key
(name_preferred_honorific_id) references particles(particle_id)
alter table NAMES
add constraint fk_name_given_name_particle_id foreign key
(name_given_name_particle_id) references particles(particle_id)
alter table NAMES
add constraint fk_name_family_name_particle_id foreign key
(name_family_name_particle_id) references particles(particle_id)
alter table NAMES
add constraint fk_name_tribe_or_clan_particle_id foreign key
(name_tribe_or_clan_particle_id) references particles(particle_id)
```

```
alter table NAMES
add constraint fk_name_legal_alias_particle_id foreign key
(name_legal_alias_particle_id) references particles(particle_id)
alter table NAMES
add constraint fk_name_use_name_particle_id foreign key (name_use_name_particle_id)
references particles(particle_id)

alter table particles
add constraint fk_particles_particle_locale_id foreign key (particle_locale_id)
references locales(locale_id)


alter table particle_orders
add constraint fk_particle_orders_particle_order_particle_id foreign key
(particle_order_particle_id) references particles(particle_id)
alter table particle_orders
add constraint fk_particle_orders_particle_order_locale_id foreign key
(particle_order_locale_id) references locales(locale_id)
alter table particle_orders
add constraint fk_particle_orders_particle_order_name_id foreign key
(particle_order_name_id) references names(name_id)

-- Initial Data

insert into locales (locale_language, locale_country)
values
('eng', 'us'),
('kat', 'ge');

go
create function get_locale_id (@language varchar(4), @country varchar(4))
returns int
begin
    declare @locale_id int;
    select @locale_id = l.locale_id
        from locales as l
        where l.locale_language = @language and l.locale_country = @country;
    return @locale_id;
end;

go

insert into particle_types (particle_type_type)
values
('Given'),
('Family'),
('Nickname'),
('Legal Alias'),
('Tribe or Clan'),
('Prefix Title '),
('Suffix'),
('Suffix Title');

go
create function get_particle_type_id (@type varchar(50))
returns int
begin
    declare @particle_type_id int;
    select @particle_type_id = l.particle_type_id
```

```
        from particle_types as l
        where l.particle_type_type = @type;
    return @particle_type_id;
end;
go


create function get_particle_id (@locale_id int, @particle_type_id int, @unicode
nvarchar(50))
returns int
begin
    declare @particle_id int;
    select @particle_id = p.particle_id
        from particles as p
        where p.particle_locale_id = @locale_id
            and p.particle_type_id = @particle_type_id
            and p.particle_unicode = @unicode;
    return @particle_id;
end;
go

create function get_particle_id_by_latin1 (@locale_id int, @particle_type_id int,
@latin1 varchar(50))
returns int
begin
    declare @particle_id int;
    select @particle_id = p.particle_id
        from particles as p
        where p.particle_locale_id = @locale_id
            and p.particle_type_id = @particle_type_id
            and p.particle_latin1 = @latin1;
    return @particle_id;
end;
GO

create function get_person_id(@email varchar(50))
returns INT
begin
    declare @person_id int;
    select @person_id = p.person_id
        from persons as p
        where p.person_email = @email;
    return @person_id;
end;
GO




-- ************************ --
-- upsert person record
-- ************************ --


DROP PROCEDURE IF EXISTS p_upsert_person_id;
GO

CREATE PROCEDURE p_upsert_person_id (
    @upsert_email_address AS VARCHAR(50)
```

```
)
AS BEGIN
    BEGIN TRANSACTION
        BEGIN TRY
            DECLARE @person_id int = NULL;

            set @person_id = (select p.person_id from persons p where
p.person_email=trim(lower(@upsert_email_address)));

            if (@person_id is NULL)
                BEGIN
                    INSERT INTO Persons (person_email) VALUES
(@upsert_email_address);
                    SET @person_id = SCOPE_IDENTITY();
                END
            COMMIT

        END TRY
        BEGIN CATCH
            ROLLBACK
            ;
            THROW 51105, 'p_upsert_person_id: An Error occurred when upserting
person.',1
        END CATCH
        RETURN @person_id
    END;
GO


-- ************************ --
-- upsert particles
-- ************************ --


DROP PROCEDURE IF EXISTS p_upsert_particle;
GO

CREATE PROCEDURE p_upsert_particle (
    @upsert_particle_type AS VARCHAR(50) ,
    @upsert_particle_unicode_text as NVARCHAR(50),
    @upsert_particle_latin1_text as VARCHAR(50),
    @upsert_particle_ipa_text as NVARCHAR(50),
    @upsert_locale_id as INT
)
AS BEGIN
    BEGIN TRANSACTION
        BEGIN TRY
            DECLARE
                @upsert_particle_id int = NULL,
                @type_id int = dbo.get_particle_type_id(@upsert_particle_type);

            set @upsert_particle_id = (
                select p.particle_id from particles p
                where p.particle_unicode = @upsert_particle_unicode_text
                and p.particle_locale_id = @upsert_locale_id
                and p.particle_type_id = @type_id)

            if(@upsert_particle_id is NULL)
                BEGIN
```

```
                    INSERT INTO particles (
                        particle_type_id,
                        particle_unicode,
                        particle_latin1,
                        particle_ipa,
                        particle_locale_id)
                    VALUES(
                        @type_id,
                        @upsert_particle_unicode_text,
                        @upsert_particle_latin1_text,
                        @upsert_particle_ipa_text,
                        @upsert_locale_id)
                    SET @upsert_particle_id = SCOPE_IDENTITY();
                END
            ELSE
                BEGIN
                    if @upsert_particle_latin1_text is not NULL
                        UPDATE particles
                        SET particle_latin1 = @upsert_particle_latin1_text
                        WHERE particle_id = @upsert_particle_id;
                    if @upsert_particle_ipa_text is not NULL
                        UPDATE particles
                        SET  particle_ipa = @upsert_particle_ipa_text
                        WHERE particle_id = @upsert_particle_id;
                END
            COMMIT
        END TRY
        BEGIN CATCH
            ROLLBACK
            ;
            THROW 51110, 'p_upsert_particle: An Error occurred when upserting a
particle.',1
        END CATCH
    RETURN @upsert_particle_id
    END;
GO


-- ************************ --
-- upsert particle orders
-- ************************ --

DROP PROCEDURE IF EXISTS p_upsert_particle_order;
GO

CREATE PROCEDURE p_upsert_particle_order (
    @upsert_particle_order_order AS INT,
    @upsert_particle_order_name_id AS INT,
    @upsert_particle_order_type AS VARCHAR(50),
    @upsert_particle_order_unicode_text as NVARCHAR(50),
    @upsert_particle_order_latin1_text as VARCHAR(50),
    @upsert_particle_order_ipa_text as NVARCHAR(50),
    @upsert_particle_order_locale_id as INT
)
AS BEGIN
    BEGIN TRANSACTION
        BEGIN TRY
            DECLARE
                @upsert_particle_order_id int = NULL,
```

```
                @_particle_id AS INT;

            set @upsert_particle_order_id = (select po.particle_order_id from
particle_orders po where po.particle_order_name_id = @upsert_particle_order_name_id
AND po.particle_order_order = @upsert_particle_order_order)

            EXEC @_particle_id = dbo.p_upsert_particle
            @upsert_particle_type = @upsert_particle_order_type,
            @upsert_particle_unicode_text = @upsert_particle_order_unicode_text,
            @upsert_particle_latin1_text = @upsert_particle_order_latin1_text,
            @upsert_particle_ipa_text =  @upsert_particle_order_ipa_text,
            @upsert_locale_id = @upsert_particle_order_locale_id;

            if(@upsert_particle_order_id is NULL)
                BEGIN
                    INSERT INTO particle_orders (particle_order_name_id,
particle_order_particle_id, particle_order_order, particle_order_locale_id)
                    VALUES(@upsert_particle_order_name_id,@_particle_id,
@upsert_particle_order_order, @upsert_particle_order_locale_id)
                    SET @upsert_particle_order_id = SCOPE_IDENTITY();
                END
            ELSE
                BEGIN
                    Update particle_orders
                    SET particle_order_particle_id = @_particle_id,
                    particle_order_locale_id = @upsert_particle_order_locale_id
                    WHERE particle_order_name_id = @upsert_particle_order_name_id
AND particle_order_order = @upsert_particle_order_order;
                END
            COMMIT
        END TRY
        BEGIN CATCH
            ROLLBACK
            ;
            THROW 51110, 'p_upsert_particle_orders: An Error occurred when
upserting particle order.',1
        END CATCH
    RETURN @upsert_particle_order_id
    END;
GO




-- ************************ --
-- DROP THE PROCEDURE TO CREATE A NAME
-- this is done first as the procedure references the custom type OrderedParticles
-- ************************ --

DROP PROCEDURE IF EXISTS p_create_name;
GO


-- ************************ --
-- DROP THE CUSTOM TYPE ORDEREDPARTICLES
-- ************************ --


DROP TYPE IF EXISTS OrderedParticles;
```

```
GO



-- ************************ --
-- CREATE a custom table type to organize all submitted name particles and their
orders
-- ************************ --

CREATE TYPE OrderedParticles AS TABLE ( particle_order_order INT NOT NULL,
particle_unicode NVARCHAR(50) NOT NULL, particle_latin1 VARCHAR(50), particle_ipa
NVARCHAR(50), particle_type_type varchar(50));
GO



-- ************************ --
-- DEFINE THE STORED PROCEDURE TO CREATE A NAME
-- ************************ --

CREATE PROCEDURE p_create_name
    (
    -- Define paramaters passed into stored procedure from application UI
    -- Default optional paramaters to NULL

    -- ****** CORE NAME INFORMATION ****** --
    @UL AS OrderedParticles READONLY,                       -- *required
    @locale_country AS VARCHAR(4),                             -- *required
    @locale_language AS VARCHAR(4),                            -- *required
    @email_address AS VARCHAR(50),                              -- *required

    @given_name_unicode AS NVARCHAR(50),                        -- *required.
delimied list.
    @given_name_latin1 AS VARCHAR(50) = NULL,                   -- optional.
delimied list.
    @given_name_ipa AS NVARCHAR(50) = NULL,                     -- optional.
delimied list.

    @family_name_unicode as NVARCHAR(50) = NULL,                -- *required.
    @family_name_latin1 as VARCHAR(50) = NULL,                  -- optional
    @family_name_ipa AS NVARCHAR(50) = NULL,                    -- optional

    @is_dead_name as BIT,                                   -- *required
    @use_name_unicode AS NVARCHAR(50) = NULL,                   -- optional.
    @use_name_latin1 AS VARCHAR(50) = NULL,                     -- optional.
    @use_name_ipa AS NVARCHAR(50) = NULL,                       -- optional.

    @is_legal_name as BIT,                                  -- *required
    @legal_alias_unicode AS NVARCHAR(50) = NULL,                -- optional.
    @legal_alias_latin1 AS VARCHAR(50) = NULL,                  -- optional.
    @legal_alias_ipa AS NVARCHAR(50) = NULL,                    -- optional.

    @preferred_locale_country as VARCHAR(4) = NULL,             -- optional.
    @preferred_locale_language as VARCHAR(4) = NULL,            -- optional.

    -- ****** GENDER IDENTITY INFORMATION ****** --
    @gender_identity AS VARCHAR(10) = NULL,                     -- optional.

    @pronoun_nominative AS NVARCHAR(50) = NULL,                  -- optional.
```

```
    @pronoun_accusative AS NVARCHAR(50) = NULL,                  -- optional.
    @pronoun_genative AS NVARCHAR(50) = NULL,                    -- optional.

    @preferred_honorific_unicode as NVARCHAR(50) = NULL,         -- optional
    @preferred_honorific_latin1 as VARCHAR(50) = NULL,           -- optional
    @preferred_honorific_ipa AS NVARCHAR(50) = NULL,             -- optional

    -- ****** CULTERAL IDENTITY INFORMATION ****** --
    @tribe_clan_name_unicode  AS NVARCHAR(50) = NULL,           -- optional.
    @tribe_clan_name_latin1 as VARCHAR(50) = NULL,              -- optional.
    @tribe_clan_name_ipa AS NVARCHAR(50) = NULL,                 -- optional.

    @mother_email as VARCHAR(50) = NULL,                        -- optional.
    @father_email as VARCHAR(50) = NULL,                        -- optional.


    -- ****** NAME OVERRIDE INFORMATION ****** --
    @override_full_name_unicode as NVARCHAR(MAX) = NULL,         -- optional.
    @override_full_name_latin1 as VARCHAR(MAX) = NULL,           -- optional.
    @override_full_name_ipa AS NVARCHAR(MAX) = NULL               -- optional.

)
AS BEGIN
    BEGIN TRY
        BEGIN TRANSACTION
            -- Orchestrate proper order of operations for inserting a name

            -- Step 1: upsert particles
            DECLARE
                @UL_Cursor CURSOR,
                @Rows INTEGER,
                @_particle_unicode NVARCHAR(50),
                @_particle_latin1 VARCHAR(50),
                @_particle_ipa NVARCHAR(50),
                @_particle_type_type VARCHAR(50),
                @_locale_id INT;

            SET @_locale_id = dbo.get_locale_id(@locale_language, @locale_country);

            SET @UL_Cursor = CURSOR FORWARD_ONLY STATIC READ_ONLY FOR
                Select
particle_unicode,particle_latin1,particle_ipa,particle_type_type from @UL;

            OPEN @UL_Cursor;

            SET @Rows = @@CURSOR_ROWS;

            While @Rows > 0
            BEGIN
                FETCH NEXT FROM @UL_Cursor INTO
                 @_particle_unicode,  @_particle_latin1,  @_particle_ipa,
@_particle_type_type

                EXEC p_upsert_particle
                @upsert_particle_type = @_particle_type_type,
                @upsert_particle_unicode_text = @_particle_unicode,
                @upsert_particle_latin1_text = @_particle_latin1,
                @upsert_particle_ipa_text =  @_particle_ipa,
                @upsert_locale_id = @_locale_id;
```

```
    SET @Rows -=1;
END;


DECLARE
    @given_name_particle_id INT,
    @family_name_particle_id INT = NULL,
    @preferred_honorific_particle_id INT = NULL,
    @tribe_or_clan_particle_id INT = NULL,
    @legal_alias_particle_id INT = NULL,
    @use_name_particle_id INT = NULL,
    @preferred_locale_id INT = NULL,
    @mother_person_id INT = NULL,
    @father_person_id INT = NULL;

    EXEC @given_name_particle_id = dbo.p_upsert_particle
    @upsert_particle_type = 'Given',
    @upsert_particle_unicode_text = @given_name_unicode,
    @upsert_particle_latin1_text = @given_name_latin1,
    @upsert_particle_ipa_text =  @given_name_ipa,
    @upsert_locale_id = @_locale_id;

    if (@family_name_unicode is NOT NULL)
    BEGIN
        EXEC @family_name_particle_id = dbo.p_upsert_particle
        @upsert_particle_type = 'Family',
        @upsert_particle_unicode_text = @family_name_unicode,
        @upsert_particle_latin1_text = @family_name_latin1,
        @upsert_particle_ipa_text =  @family_name_ipa,
        @upsert_locale_id = @_locale_id;
    END;

    if (@preferred_honorific_unicode is NOT NULL)
    BEGIN
        EXEC @preferred_honorific_particle_id = dbo.p_upsert_particle
        @upsert_particle_type = 'Prefix Title',
        @upsert_particle_unicode_text = @preferred_honorific_unicode,
        @upsert_particle_latin1_text = @preferred_honorific_latin1,
        @upsert_particle_ipa_text =  @preferred_honorific_ipa,
        @upsert_locale_id = @_locale_id;
    END;

    if (@tribe_clan_name_unicode is NOT NULL)
    BEGIN
        EXEC @tribe_or_clan_particle_id = dbo.p_upsert_particle
        @upsert_particle_type = 'Tribe or Clan',
        @upsert_particle_unicode_text = @tribe_clan_name_unicode,
        @upsert_particle_latin1_text = @tribe_clan_name_latin1,
        @upsert_particle_ipa_text =  @tribe_clan_name_ipa,
        @upsert_locale_id = @_locale_id;
    END;


    if (@legal_alias_unicode is NOT NULL)
    BEGIN
        EXEC @legal_alias_particle_id = dbo.p_upsert_particle
        @upsert_particle_type = 'Legal Alias',
        @upsert_particle_unicode_text = @legal_alias_unicode,
```

```
                    @upsert_particle_latin1_text = @legal_alias_latin1,
                    @upsert_particle_ipa_text =  @legal_alias_ipa,
                    @upsert_locale_id = @_locale_id;
                END;


                if (@use_name_unicode is NOT NULL)
                BEGIN
                    EXEC @use_name_particle_id = dbo.p_upsert_particle
                    @upsert_particle_type = 'Nickname',
                    @upsert_particle_unicode_text = @use_name_unicode,
                    @upsert_particle_latin1_text = @use_name_latin1,
                    @upsert_particle_ipa_text =  @use_name_ipa,
                    @upsert_locale_id = @_locale_id;
                END;

                if (@mother_email is not null)
                BEGIN
                    SET @mother_person_id = dbo.get_person_id(@mother_email);
                END;

                if (@father_email is NOT NULL)
                BEGIN
                    SET @father_person_id = dbo.get_person_id(@father_email);
                END;

                if (@preferred_locale_country is not null)
                BEGIN
                    SET  @preferred_locale_id =
dbo.get_locale_id(@preferred_locale_language, @preferred_locale_country);
                END;


            -- Step 2: upsert person.
            DECLARE @person_id int
            EXEC @person_id = DBO.p_upsert_person_id @upsert_email_address =
@email_address;


            -- Step 3: insert into names
                INSERT INTO dbo.Names (
                    name_locale_id,
                    name_is_legal_name,
                    name_is_dead_name,
                    name_preferred_locale_id,
                    name_gender_identity,
                    name_preferred_honorific_id,
                    name_preferred_pronoun_nominative,
                    name_preferred_pronoun_accusative,
                    name_preferred_pronoun_genative,
                    name_override_full_name,
                    name_override_full_name_latin1,
                    name_override_full_name_ipa,
                    name_given_name_particle_id,
                    name_family_name_particle_id,
                    name_tribe_or_clan_particle_id,
                    name_legal_alias_particle_id,
                    name_use_name_particle_id,
                    name_person_id,
```

```
                    name_mother_person_id,
                    name_father_person_id)
                values(
                    @_locale_id,
                    @is_legal_name,
                    @is_dead_name,
                    @preferred_locale_id,
                    @gender_identity,
                    @preferred_honorific_particle_id,
                    @pronoun_nominative,
                    @pronoun_accusative,
                    @pronoun_genative,
                    @override_full_name_unicode,
                    @override_full_name_latin1,
                    @override_full_name_ipa,
                    @given_name_particle_id,
                    @family_name_particle_id,
                    @tribe_or_clan_particle_id,
                    @legal_alias_particle_id,
                    @use_name_particle_id,
                    @person_id,
                    @mother_person_id,
                    @father_person_id
                );
                DECLARE @name_id INT = SCOPE_IDENTITY();

            -- Step 3: insert into particle_order for each particle ID
            DECLARE
                @_particle_order INT,
                @_particle_id INT;

            DELETE FROM dbo.particle_orders where particle_order_name_id =
@name_id;

            SET @UL_Cursor = CURSOR FORWARD_ONLY STATIC READ_ONLY FOR
                Select particle_order_order, particle_unicode, particle_latin1,
particle_ipa, particle_type_type from @UL;

            OPEN @UL_Cursor;

            SET @Rows = @@CURSOR_ROWS;

            While @Rows > 0
            BEGIN
                FETCH NEXT FROM @UL_Cursor INTO
                @_particle_order, @_particle_unicode,  @_particle_latin1,
@_particle_ipa, @_particle_type_type

                EXEC p_upsert_particle_order
                    @upsert_particle_order_order = @_particle_order,
                    @upsert_particle_order_name_id = @name_id,
                    @upsert_particle_order_type = @_particle_type_type,
                    @upsert_particle_order_unicode_text = @_particle_unicode,
                    @upsert_particle_order_latin1_text = @_particle_latin1,
                    @upsert_particle_order_ipa_text =  @_particle_ipa,
                    @upsert_particle_order_locale_id = @_locale_id;

                SET @Rows -=1;
            END;
```

```
        COMMIT
    END TRY
    BEGIN CATCH
        ROLLBACK
        ;
        --THROW 51120, 'p_create_name: An Error occurred when attempting to insert
a new name.',1
        THROW
    END CATCH
END;
GO


-- ************************ --
-- PERFORM THE INSERTS
-- This secton simulates the submission of values from the front-end form
-- ************************ --
GO
DECLARE @Name1 OrderedParticles;

INSERT @Name1 VALUES
    (1, 'Mr.', 'mister', N'ˈmɪstəɹ', 'Prefix Title'),
    (2, 'Christopher', 'Christopher', N'ˈkrɪs.tə.fər', 'Given'),
    (3, 'Alan', 'Alan', N'álən', 'Given'),
    (4, 'Murphy', 'Murphy', N'ˈmɝfi', 'Family'),
    (5, 'Jr.', 'junior', N'ˈd͡ʒuniɚ', 'Suffix');

EXEC dbo.p_create_name @UL = @Name1, @locale_country = 'us',
@locale_language='eng', @email_address='cmurph66@syr.edu',
    @use_name_unicode = N'Chris', @use_name_latin1='Chris', @use_name_ipa='ˈkrɪs',
    @given_name_unicode='Christopher', @family_name_unicode='Murphy',
@is_dead_name=0, @is_legal_name=1,
    @preferred_honorific_unicode='Mr.';


DECLARE @Name2 OrderedParticles;

INSERT @Name2 VALUES
    (1, N'Dr.', 'doctor', N'dɔ́ktər', 'Prefix Title'),
    (2, N'La Monte', 'La Monte', N'ləˈmɒnt', 'Given'),
    (3, N'Henry', 'Henry', N'ˈhe̩nry', 'Given'),
    (4, N'Piggy', 'Piggy', N'pɪ́gɪj', 'Given'),
    (5, N'Yarroll', 'Yarroll', N'ˈjæɹəl', 'Family'),
    (6, N'esq.', 'esquire', N'ɪsgwɑ́jə', 'Suffix Title');

EXEC dbo.p_create_name @UL = @Name2, @locale_country = 'us',
@locale_language='eng',
    @email_address='piggy@cmu.edu', @given_name_unicode=N'La Monte',
@family_name_unicode=N'Yarroll', @is_dead_name=0, @is_legal_name=1,
    @preferred_honorific_unicode=N'Dr.';


DECLARE @Name3 OrderedParticles;

INSERT @Name3 VALUES
    (1, 'Miss', 'Miss', N'mɪ́s', 'Prefix Title'),
    (2, 'Eve', 'Eve', N'iːv', 'Given'),
    (3, 'Karenina', 'Karenina', N'kɐˈrʲenʲɪnə', 'Given'),
```

```
    (4, 'Prastein', 'Prastein', N'prɑ́ːstin', 'Family');

EXEC dbo.p_create_name @UL = @Name3, @locale_country = 'us',
@locale_language='eng',
    @email_address='baqaqi@gmail.com', @given_name_unicode=N'Eve',
@family_name_unicode=N'Prastein', @is_dead_name=1, @is_legal_name=0,
    @preferred_honorific_unicode=N'Miss';


DECLARE @Name4 OrderedParticles;

INSERT @Name4 VALUES
    (1, N'Mrs.', 'missus', N'ˈmɪsəz', 'Prefix Title'),
    (2, N'Eve', 'Eve', N'iːv', 'Given'),
    (3, N'Karenina', 'Karenina', N'kɐˈrʲenʲɪnə', 'Given'),
    (4, N'Yarroll', 'Yarroll',  NULL, 'Family');

EXEC dbo.p_create_name @UL = @Name4, @locale_country = 'us',
@locale_language='eng',
    @email_address='baqaqi@gmail.com', @given_name_unicode=N'Eve',
@family_name_unicode=N'Yarroll', @is_dead_name=0, @is_legal_name=1,
    @preferred_honorific_unicode=N'Mrs.';


DECLARE @Name5 OrderedParticles;

INSERT @Name5 VALUES
    (1, N'ბატონ','baton', N'baton','Prefix Title'),
    (2, N'ლამონტი','lamonti', N'laˈmɒnti', 'Given'),
    (3, N'ჰენრი','henri', N'ˈhenri','Given'),
    (4, N'პიგი', 'pigi', N'ˈpigi','Given'),
    (5, N'იაროლი', 'iaroli', N'iaroli', 'Family');

EXEC dbo.p_create_name @UL = @Name5, @locale_country = 'ge',
@locale_language='kat',
    @email_address='piggy@cmu.edu', @given_name_unicode=N'ლამონტი',
@family_name_unicode=N'იაროლი', @is_dead_name=0, @is_legal_name=0,
    @preferred_honorific_unicode= N'ბატონ';

DECLARE @Name6 OrderedParticles;

INSERT @Name6 VALUES
    (1, N'Mr.', 'mister', N'ˈmɪstəɹ', 'Prefix Title'),
    (2, N'Oluwaseyi', 'Oluwaseyi', N'oluwäʃɛ́jɪ', 'Given'),
    (3, N'Olufemi', 'Olufemi', N'olufɛ́mɪ', 'Given'),
    (4, N'Durosinmi-Etti', 'Durosinmi-Etti', N'dʊəroʃinmi ɛ́tɪj', 'Family');

EXEC dbo.p_create_name @UL = @Name6, @locale_country = 'us',
@locale_language='eng', @email_address='odurosin@syr.edu',
    @use_name_unicode = N'Seyi', @use_name_latin1='Seyi', @use_name_ipa=N'ʃɛ́jɪ',
    @given_name_unicode='Oluwaseyi', @family_name_unicode='Durosinmi-Etti',
@is_dead_name=0, @is_legal_name=1,
    @preferred_honorific_unicode='Mr.';

DECLARE @Name7 OrderedParticles;

INSERT @Name7 VALUES
    (1, N'Mr.', 'mister', N'ˈmɪstəɹ', 'Prefix Title'),
    (2, N'Patrick', 'Patrick', N'pátrɪk', 'Given'),
```

```
    (3, N'Le', 'Le', N'láj', 'Family');

EXEC dbo.p_create_name @UL = @Name7, @locale_country = 'us',
@locale_language='eng', @email_address='ple107@syr.edu',
    @use_name_unicode = N'Pat', @use_name_latin1='Pat', @use_name_ipa=N'pát',
    @given_name_unicode=N'Patrick', @family_name_unicode='Le', @is_dead_name=0,
@is_legal_name=1,
    @preferred_honorific_unicode='Mr.';


GO

-- ************************ --
-- VALIDATE THE TEST DATA
-- ************************ --
select * from names;

select * from particles;

select * from persons;

select * from particle_orders;

select dbo.get_locale_id('eng', 'us') as eng_us_locale;

select dbo.get_particle_type_id('Family') as family_particle_type;

select dbo.get_particle_id(
    dbo.get_locale_id('eng', 'us'),
    dbo.get_particle_type_id('Given'),
    'La Monte'
) as la_monte_given;

select dbo.get_particle_id_by_latin1(
    dbo.get_locale_id('kat', 'ge'),
    dbo.get_particle_type_id('Given'),
    'lamonti'
) as la_monte_given_kat;



-- ************************ --
-- CREATE THE VIEWS AND SUPORTING FUNCTIONS NEEDED TO CREATE THE VIEWS
-- ************************ --
drop view if exists v_combined
GO
create view v_combined as (
    select *
    from names
    join persons on person_id = name_person_id
    join particle_orders on particle_order_name_id = name_id
    join particles on particle_id = particle_order_particle_id
)

GO
drop function if exists get_particle_type;
drop function if exists get_legal_names;
drop function if exists get_full_names;
drop function if exists get_initials;
```

```
drop function if exists get_informal_name;
drop function if exists get_formal_name;
drop function if exists get_use_name;
GO

create function get_particle_type(@particle_type_id int)
returns varchar(50) AS
begin
    declare @retval varchar(50);
    select @retval = particle_type_type
    from particle_types
    where particle_type_id = @particle_type_id;
    return @retval
end;
GO
drop function if exists get_use_name
GO

create function get_use_name()
returns table as
    return select name_person_id as person_id, name_id, name_locale_id,
name_is_dead_name as is_dead_name, person_email,
    isnull(u.particle_unicode, g.particle_unicode) as use_name_unicode,
    isnull(u.particle_latin1, g.particle_latin1) as use_name_latin1,
    isnull(u.particle_ipa, g.particle_ipa) as use_name_ipa
    from names
    join persons on name_person_id = person_id
    left join particles as u on name_use_name_particle_id = u.particle_id
    left join particles as g on name_given_name_particle_id = g.particle_id
GO

GO

create function get_formal_name()
returns table AS
    return select person_id, name_id, name_locale_id,
    particle_unicode as formal_name_unicode,
    particle_latin1 as formal_name_latin1,
    particle_ipa as formal_name_ipa,
    name_is_dead_name as is_dead_name
    from v_combined
    where dbo.get_particle_type(particle_type_id) in ('Family')
GO

create function get_informal_name()
returns table AS
    return select person_id, name_id, name_locale_id,
    name_is_dead_name as is_dead_name,
    particle_unicode as informal_name_unicode,
    particle_latin1 as informal_name_latin1,
    particle_ipa as informal_name_ipa
    from (
        SELECT person_id, name_id, name_locale_id,
      particle_unicode, particle_latin1, particle_ipa,
      ROW_NUMBER() OVER (PARTITION BY name_id ORDER BY name_id) AS row_num,
      name_is_dead_name
        FROM v_combined
        where dbo.get_particle_type(particle_type_id) in ('Given')) t
    where t.row_num = 1
```

```
GO

create function get_legal_names ()
returns table AS
    return select person_id, name_id, name_locale_id,
    STRING_AGG(particle_unicode, ' ') within group (order by person_id, name_id,
particle_order_id) as legal_name_unicode,
    STRING_AGG(particle_ipa, ' ') within group (order by person_id, name_id,
particle_order_id) as legal_name_ipa,
    STRING_AGG(particle_latin1, ' ') within group (order by person_id, name_id,
particle_order_id) as legal_name_latin1,
    name_is_dead_name as is_dead_name
    from v_combined
    where name_is_legal_name = 1 and dbo.get_particle_type(particle_type_id) in
('Given', 'Family', 'Suffix')
    group by person_id, name_id, name_locale_id, name_is_dead_name

GO

create function get_full_names()
returns table AS
    return select person_id, name_id, name_locale_id, name_is_dead_name as
is_dead_name, person_email,
    STRING_AGG(particle_unicode, ' ') within group (order by person_id, name_id,
particle_order_id) as full_name_unicode,
    STRING_AGG(particle_ipa, ' ') within group (order by person_id, name_id,
particle_order_id) as full_name_ipa,
    STRING_AGG(particle_latin1, ' ') within group (order by person_id, name_id,
particle_order_id) as full_name_latin1
    from v_combined
    group by person_id, name_id, name_locale_id, name_is_dead_name, person_email
GO
drop function if exists get_honorific
GO
create function get_honorific()
returns table as
    return
    select name_person_id as person_id, name_id, name_locale_id, name_is_dead_name
as is_dead_name, person_email,
    h.particle_unicode as honorific_unicode,
    h.particle_latin1 as honorific_latin1,
    h.particle_ipa as honorific_ipa
    from [names]
    join persons on name_person_id = person_id
    join particles as h on h.particle_id = name_preferred_honorific_id
GO
drop function if exists get_family_name
GO
create function get_family_name()
returns table as
    return select name_person_id as person_id, name_id, name_locale_id,
name_is_dead_name as is_dead_name, person_email,
    f.particle_unicode as family_name_unicode,
    f.particle_latin1 as family_name_latin1,
    f.particle_ipa as family_name_ipa
    from names
    join persons on name_person_id = person_id
    join particles as f on name_family_name_particle_id = f.particle_id
```

```
GO
create function get_initials()
returns table AS
    return select person_id, name_id, name_locale_id, name_is_dead_name as
is_dead_name,
    STRING_AGG(LEFT(particle_unicode, 1), '') within group (order by
person_id,name_id, particle_order_id) as initials_unicode,
    STRING_AGG(LEFT(particle_latin1, 1), '') within group (order by
person_id,name_id,particle_order_id) as initials_latin1
    from v_combined
    where dbo.get_particle_type(particle_type_id) in ('Given', 'Family')
    group by person_id, name_id, name_locale_id, name_is_dead_name
GO

drop view if exists v_eng_us;
GO

create view v_eng_us as (
    select  f.person_id, f.name_id, f.person_email, f.name_locale_id,
f.is_dead_name,
    full_name_unicode, full_name_latin1, full_name_ipa,
    legal_name_unicode, legal_name_latin1, legal_name_ipa,
    formal_name_unicode,
    formal_name_latin1,
    formal_name_ipa,
    honorific_unicode + ' ' + family_name_unicode as short_formal_unicode,
    honorific_latin1 + ' ' + family_name_latin1 as short_formal_latin1,
    honorific_ipa + ' ' + family_name_ipa as short_formal_ipa,
    initials_unicode, initials_latin1,
    use_name_unicode,
    use_name_latin1,
    use_name_ipa
    from dbo.get_full_names() f
    left join dbo.get_legal_names() l on f.name_id = l.name_id
    left join dbo.get_initials() i on f.name_id = i.name_id
    left join dbo.get_informal_name() ifn on f.name_id = ifn.name_id
    left join dbo.get_formal_name() fn on f.name_id = fn.name_id
    left join dbo.get_use_name() un on f.name_id = un.name_id
    left join dbo.get_honorific() ho on f.name_id = ho.name_id
    left join dbo.get_family_name() fa on f.name_id = fa.name_id
    where f.name_locale_id = dbo.get_locale_id('eng', 'us')
    )
GO
select * from dbo.get_family_name()
drop view if exists v_kat_ge;
GO

create view v_kat_ge as (
    select  f.person_id, f.name_id, f.person_email, f.name_locale_id,
f.is_dead_name,
    full_name_unicode, full_name_latin1, full_name_ipa,
    legal_name_unicode, legal_name_latin1, legal_name_ipa,
    formal_name_unicode,
    formal_name_latin1,
    formal_name_ipa,
    honorific_unicode + ' ' + use_name_unicode as short_formal_unicode,
    honorific_latin1 + ' ' + use_name_latin1 as short_formal_latin1,
    honorific_ipa + ' ' + use_name_ipa as short_formal_ipa,
    initials_unicode, initials_latin1,
```

```
    use_name_unicode,
    use_name_latin1,
    use_name_ipa
    from dbo.get_full_names() f
    left join dbo.get_legal_names() l on f.name_id = l.name_id
    left join dbo.get_initials() i on f.name_id = i.name_id
    left join dbo.get_informal_name() ifn on f.name_id = ifn.name_id
    left join dbo.get_formal_name() fn on f.name_id = fn.name_id
    left join dbo.get_use_name() un on f.name_id = un.name_id
    left join dbo.get_honorific() ho on f.name_id = ho.name_id
    left join dbo.get_family_name() fa on f.name_id = fa.name_id
    where f.name_locale_id = dbo.get_locale_id('kat', 'ge')
    )
GO

-- ************************ --
-- VALIDATE THE VIEWS AND VIEW FUNCTIONS
-- ************************ --

select * from get_legal_names();
select * from get_initials();
select * from get_full_names();
select * from get_informal_name();
select * from get_formal_name();
select * from get_use_name();
select * from get_honorific();
select * from get_family_name();
select * from v_eng_us;

GO


-- ************************ --
--  10 QUESTIONS - ENG_US LOCALE SPECIFIC
-- ************************ --


-- 1.) How many contacts (unique persons) in the database (eng_us locale)
-- used to track growth of the system
select count(distinct(person_id)) from v_eng_us;

-- 2.) Show preferred names vs. active full name (eng_us locale)
-- Who are the contacts/clients in the system?
-- used to ensure preferred naming preferences are understood in account
interactions
select person_id, use_name_unicode + ':' + formal_name_unicode as preferred_name,
full_name_unicode as complete_name from v_eng_us
where is_dead_name != 1;

-- 3.) Show use name vs. non-dead legal name for clients How many legal vs. non
legal names?
-- used to ensure legal names are used for formal documentation, agreements, etc.
select person_id, use_name_unicode + ':' + formal_name_unicode as preferred_name,
legal_name_unicode as legal_name from v_eng_us
where is_dead_name != 1;

-- 4.) Show preferred names and emails
-- How should we address contacts/clients for email marketing and system
personalization?
```

```
select person_email, use_name_unicode as perferred_given_name, use_name_unicode + '
' + formal_name_unicode as preferred_full_name from v_eng_us
where is_dead_name != 1;

-- 5.) Show full name associated with email addresses
-- How should we address contacts/clients for formal communications and engagements
requiring introductions?
select person_email, full_name_unicode from v_eng_us
where is_dead_name != 1;

-- 6.) Show legal name associated with an email address.
-- How should we address contacts/clients in formal legal documentation?
select person_email, legal_name_unicode as legal_name from v_eng_us
where is_dead_name != 1;

-- 7.) Show the full set of unique email addresses (eng_us locale).
-- reviewed for data maintenance, data cleansing, and GDPR legal compliance
select distinct(person_email) from v_eng_us;

-- 8.) Show all dead names (eng_us locale)
-- this list can be reviewed to ensure all communications and touch points with the
person avoids the use of the dead name
select person_id, name_id, person_email, full_name_unicode as "dead_name - do not
use" from v_eng_us where is_dead_name = 1;

-- 9.) How many records lack a particle IPA?
-- used for data maintenance and cleansing.  A linguist would use this query to
find records to update.
-- future versions of the application could integrate with an ipa service and auto-
populate these records
select * from particles where particle_ipa is null or particle_ipa = '';


-- 10.) getting average amount of particles per name
-- used to determine if the system is being used as intended.
-- can be used as a metric for marketing teams that maintain a % Complete for IPA
entries
-- to ensure the particles being entered assist the team in properly personalizing
-- interactions with clients.
select avg((LEN(f.full_name_unicode) - LEN(REPLACE(f.full_name_unicode, ' ', '')) +
1)) as avg_particle_count from
(select full_name_unicode from v_eng_us) f;


-- 11.) How to address the customer formally.
select person_email, short_formal_unicode
from v_eng_us;

-- ************************ --
--  10 QUESTIONS - KAT_GE LOCALE SPECIFIC
-- ************************ --
-- 11.) How to address the customer formally.
select person_email, short_formal_unicode, short_formal_latin1, short_formal_ipa
from v_kat_ge;

select * from v_kat_ge;
```