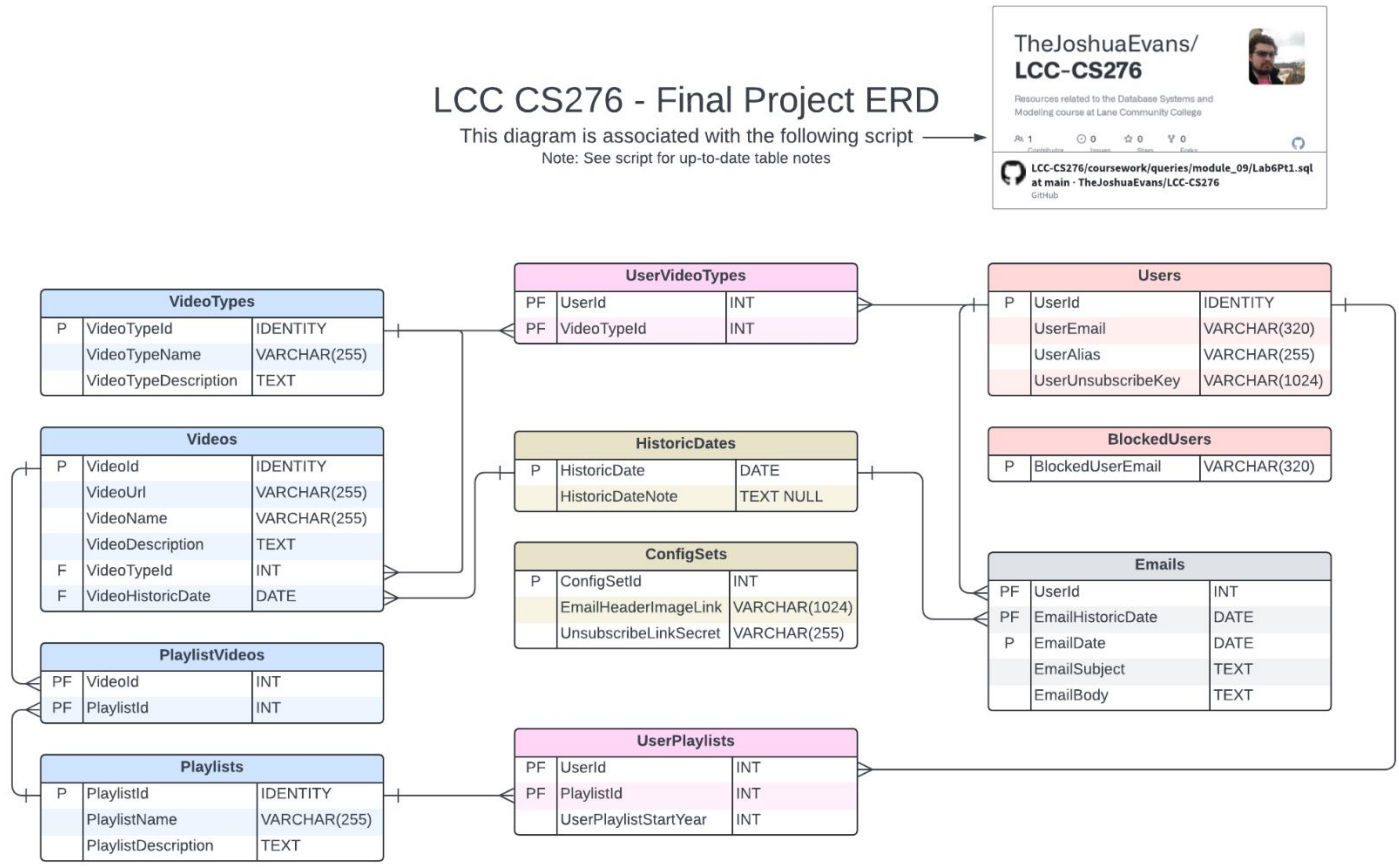
Lab 6

Joshua Evans | CS 276 | Due 2024-06-02

Part 1 – Database Creation:

The SQL script files associated with the following code snippets can be found here:

https://github.com/TheJoshuaEvans/LCC-CS276/blob/main/coursework/queries/module_09/Lab6Pt1.sql




```sql
USE master;
GO

-- These lines can cause problems when performing development since the IDE creates a
connection to the database to help with auto-complete
DROP DATABASE IF EXISTS RealTimeWW2;
CREATE DATABASE RealTimeWW2;
GO

USE [RealTimeWW2];

-- Because of the complex relationship between tables, it's best to delete them all up front
-- then create them together
DROP TABLE IF EXISTS [Emails];
DROP TABLE IF EXISTS [ConfigSets];
```

```sql
DROP TABLE IF EXISTS [UserVideoTypes];
DROP TABLE IF EXISTS [UserPlaylists];
DROP TABLE IF EXISTS [Users];
DROP TABLE IF EXISTS [BlockedUsers];
DROP TABLE IF EXISTS [PlaylistVideos];
DROP TABLE IF EXISTS [Playlists];
DROP TABLE IF EXISTS [Videos];
DROP TABLE IF EXISTS [VideoTypes];
DROP TABLE IF EXISTS [HistoricDates];

/* All the dates of the war with content, starting with 1939-09-01 */
CREATE TABLE [HistoricDates](
  -- The actual date
  HistoricDate DATE NOT NULL PRIMARY KEY,
  -- Note that gives some context for the historic date
  HistoricDateNote TEXT NULL
);

/* Static values used by the system to perform various actions */
CREATE TABLE [ConfigSets](
  -- ID of the config set. 0 is the "main" config
  ConfigSetId INT NOT NULL IDENTITY(0,1) PRIMARY KEY,
  -- URL of the header image used in the generated emails
  EmailHeaderImageLink VARCHAR(1024) NOT NULL,
  -- Secret passphrase used to generate the unsubscribe key for users
  UnsubscribeLinkSecret VARCHAR(255) NOT NULL
)

/* Users that have subscribed to the service */
CREATE TABLE [Users](
  -- Unique ID of the user
  UserId INT IDENTITY(0,1) NOT NULL PRIMARY KEY,
  -- The user's email address. Must be unique
  UserEmail VARCHAR(320) NOT NULL UNIQUE,
  -- Name the user wishes to be called in their emails
  UserAlias VARCHAR(1024) NOT NULL,
  -- Key used in the unsubscribe link in the user's emails. Generated using the user's email
  -- address and the secret from the config set
  UserUnsubscribeKey VARCHAR(1024) NOT NULL
);

/*
  List of users that have specifically requested not to receive emails from the service (not
  just unsubscribed). Attempting to create a user with a blocked user email will cause an
error
*/
CREATE TABLE [BlockedUsers](
  -- Email address that has been blocked
  BlockedUserEmail VARCHAR(320) NOT NULL PRIMARY KEY
);
```

```sql
/*
  Videos are split into playlists that span a certain period of time and focus on specific
  topics. Most commonly used will be the "main" playlist that includes all videos, but other
  playlists will be available for things like "the siege of Berlin" or "Stalingrad". These
  playlists aren't necessarily related to the playlists on the YouTube channel
*/
CREATE TABLE [Playlists](
  -- Unique ID of the playlist
  PlaylistId INT IDENTITY(0,1) NOT NULL PRIMARY KEY,
  -- Human friendly name of the playlist
  PlaylistName VARCHAR(255) NOT NULL,
  -- Short human friendly description of the playlist
  PlaylistDescription TEXT NOT NULL
);

/*
  Every video has a single "type" the corresponds to the series the video is a part of.
Examples
  include the main weekly series, War Against Humanity, Spies in Ties, Out of the Foxholes,
etc
*/
CREATE TABLE [VideoTypes](
  -- Unique ID of the video type
  VideoTypeId INT IDENTITY(0,1) NOT NULL PRIMARY KEY,
  -- Human friendly name of the video type
  VideoTypeName VARCHAR(255) NOT NULL,
  -- Short human friendly description of the video type
  VideoTypeDescription TEXT NOT NULL
);

/*
  Every video that is part of the World War 2 in real time project. This doesn't include
special
  announcements or other time sensitive content that is no longer relevant
*/
CREATE TABLE [Videos](
  -- Unique ID for the video
  VideoId INT IDENTITY(0,1) NOT NULL PRIMARY KEY,
  -- Link to YouTube where the video can be found
  VideoUrl VARCHAR(255) NOT NULL UNIQUE,

  -- Name of the video on YouTube
  VideoName VARCHAR(255) NOT NULL,
  -- The video's description on YouTube, without the copied elements
  VideoDescription TEXT,
  -- ID of the video type associated with this video
  VideoTypeId INT NOT NULL,
  -- Historic date the video is meant to take place on
  VideoHistoricDate DATE NOT NULL,
```

```
  FOREIGN KEY (VideoTypeId) REFERENCES [VideoTypes](VideoTypeId),
  FOREIGN KEY (VideoHistoricDate) REFERENCES [HistoricDates](HistoricDate)
);

/* Join table connecting playlists to videos */
CREATE TABLE [PlaylistVideos](
  -- ID of the playlist
  PlaylistId INT NOT NULL,
  -- ID of the video
  VideoId INT NOT NULL,
  PRIMARY KEY (VideoId, PlaylistId),
  FOREIGN KEY (VideoId) REFERENCES [Videos](VideoId),
  FOREIGN KEY (PlaylistId) REFERENCES [Playlists](PlaylistId)
);

/* Join table that connects users to the video types they wish to receive emails about */
CREATE TABLE [UserVideoTypes](
  -- ID of the video type the user subscribed to
  VideoTypeId INT NOT NULL,
  -- ID of the user subscribed to this type
  UserId INT NOT NULL,
  PRIMARY KEY (UserId, VideoTypeId),
  FOREIGN KEY (UserId) REFERENCES [Users](UserId),
  FOREIGN KEY (VideoTypeId) REFERENCES [VideoTypes](VideoTypeId)
)

/* Playlists that a user has subscribed to */
CREATE TABLE [UserPlaylists](
  -- ID of the playlist being subscribed to
  PlaylistId INT NOT NULL,
  -- ID of the subscribed user
  UserId INT NOT NULL,
  -- The year the user subscribed to the playlist. Normalized to the start of the war, not
the
  -- first video of the playlist
  UserPlaylistStartYear INT NOT NULL,
  PRIMARY KEY (UserId, PlaylistId),
  FOREIGN KEY (UserId) REFERENCES [Users](UserId),
  FOREIGN KEY (PlaylistId) REFERENCES [Playlists](PlaylistId)
);

/* The emails that are generated for distribution */
CREATE TABLE [Emails](
  -- ID of the user the email is sent to
  UserId INT NOT NULL,
  -- Historic date associated with the email
  EmailHistoricDate DATE NOT NULL,
  -- Date (not including time) the email was generated
  EmailDate DATE NOT NULL,
  -- Subject of the email
```

```
  EmailSubject TEXT NOT NULL,
  -- HTML Body of the email
  EmailBody TEXT NOT NULL,
  PRIMARY KEY (UserId, EmailHistoricDate, EmailDate),
  FOREIGN KEY (UserId) REFERENCES [Users](UserId),
  FOREIGN KEY (EmailHistoricDate) REFERENCES [HistoricDates](HistoricDate),
);
GO

-- -------------- INSERT STATEMENTS -------------- --
-- Much more data will be required for the actual app, this is just a functional sampling

-- Initialize the default configuration set
SET IDENTITY_INSERT [ConfigSets] ON;
INSERT INTO [ConfigSets] (ConfigSetId, EmailHeaderImageLink, UnsubscribeLinkSecret) VALUES
  (0,
'https://yt3.googleusercontent.com/ytc/AIdro_ntBfc7SOI3SmH0bOy_tyjaCY_g57TSNrGgEnH_Fp6H3w=s176-c-k-c0x00ffffff-no-rj', 'k8m1lQmwQa45DtK?5NuUJuLhXH?Ec5e?iu@pnJev');


-- Set up some initial historic dates. A date must be present for every video, but dates with no
-- content do not need to be defined
INSERT INTO [HistoricDates] (HistoricDate, HistoricDateNote) VALUES ('1939-09-01', 'The very first day of the war');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-09-08');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-09-15');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-09-22');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-09-29');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-10-06');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-10-13');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-10-20');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-10-27');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-11-03');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-11-10');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-11-17');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-11-24');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-12-01');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-12-08');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-12-15');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-12-22');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1939-12-29');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1940-01-05');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1940-01-12');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1940-01-19');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1940-01-26');
INSERT INTO [HistoricDates] (HistoricDate, HistoricDateNote) VALUES ('1940-01-28', 'The first special episode that breaks the weekly pattern is today');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1940-02-02');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1940-02-09');
```

```
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1940-02-16');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1940-02-23');
INSERT INTO [HistoricDates] (HistoricDate, HistoricDateNote) VALUES ('1940-02-28', 'This day
marks the first episode of Out of the Foxholes');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1940-03-02');
INSERT INTO [HistoricDates] (HistoricDate, HistoricDateNote) VALUES ('1940-03-05', 'Today is
the first episode of War Against Humanity, a brutal, but extremely important series');
INSERT INTO [HistoricDates] (HistoricDate) VALUES ('1940-03-09')
GO

-- Add some video types
INSERT INTO [VideoTypes] (VideoTypeName, VideoTypeDescription) VALUES ('Weekly Episodes',
'The main weekly series which originally aired every Saturday');
INSERT INTO [VideoTypes] (VideoTypeName, VideoTypeDescription) VALUES ('War Against
Humanity', 'A deep dive into the atrocities committed on all sides during this war');
INSERT INTO [VideoTypes] (VideoTypeName, VideoTypeDescription) VALUES ('Special Episodes',
'One-off special episodes about various different topics');
INSERT INTO [VideoTypes] (VideoTypeName, VideoTypeDescription) VALUES ('Out of the Foxholes',
'Q&A Episodes in which audience questions are addressed');
GO

-- Add some videos of various types
INSERT INTO [Videos] (VideoUrl, VideoName, VideoDescription, VideoTypeId, VideoHistoricDate)
VALUES
  (
    'https://youtu.be/2b7GY4BSUmU',
    '001 -The Polish-German War - WW2 - September 1, 1939 [IMPROVED]',
    'When Germany invades Poland on September 1, 1939, the world is already at the brink of a
new world war…',
    0,
    '1939-09-01'
  );
INSERT INTO [Videos] (VideoUrl, VideoName, VideoDescription, VideoTypeId, VideoHistoricDate)
VALUES
  (
    'https://youtu.be/sVPLeZ_Vpw8',
    '002 - World War Two Begins - WW2 - September 8, 1939 [IMPROVED]',
    'The German-Polish war is the match that ignites the flames that finally burn British
Prime Minister Neville Chamberlain''s appeasement efforts to the ground.',
    0,
    '1939-09-08'
  );
INSERT INTO [Videos] (VideoUrl, VideoName, VideoDescription, VideoTypeId, VideoHistoricDate)
VALUES
  (
    'https://youtu.be/dHd-2eD0ejU',
    '003 - Poland on Her Own - WW2 - September 15, 1939 [IMPROVED]',
    'When the Wehrmacht and the SS continue devastating Poland and her people in the first
weeks of September, her last chance is her western allies.',
    0,
```

```
    '1939-09-15'
  );
INSERT INTO [Videos] (VideoUrl, VideoName, VideoDescription, VideoTypeId, VideoHistoricDate)
VALUES
  (
    'https://youtu.be/8vjBp-qyNVE',
    '004 - The Russians are Coming! - The Soviet Invasion of Poland - WW2 -September 22, 1939
[IMPROVED]',
    'When the USSR crushes the plans of the Allies for Poland and the Japanese plans in China
in the same week, it is Germany that benefits.',
    0,
    '1939-09-22'
  );
INSERT INTO [Videos] (VideoUrl, VideoName, VideoDescription, VideoTypeId, VideoHistoricDate)
VALUES
  (
    'https://youtu.be/OOPiUaakSXM',
    '005 - Poland is Crushed - WW2 - 29 September, 1939 [IMPROVED]',
    'Facing two enemies at once, Poland finds itself in a crushing vice after less than a
month of war and the Polish forces must flee their own country to live to fight another
day.',
    0,
    '1939-09-29'
  );
INSERT INTO [Videos] (VideoUrl, VideoName, VideoDescription, VideoTypeId, VideoHistoricDate)
VALUES
  (
    'https://youtu.be/k1gR6LrR5P4',
    '006 - Poland Falls and China Rises - WW2 - October 6, 1939 [IMPROVED]',
    'In the West, the sun sets on Poland as the last forces surrender, but her defenders are
already regrouping abroad. In the East, the sun rises on China as Japan meets yet another
defeat',
    0,
    '1939-10-06'
  );


-- Skip ahead to some different episode types...
INSERT INTO [Videos] (VideoUrl, VideoName, VideoDescription, VideoTypeId, VideoHistoricDate)
VALUES
  (
    'https://youtu.be/Etfhio8vrXE',
    'The T-26 and Tank Warfare in Finland and China - WORLD WAR TWO Special',
    'The T-26 tank was one of the most frequently used tanks during the first battles of
World War Two. It saw action in the Soviet Union, Finland and China. In our first
collaborative video with the Tank Museum in Bovington, UK, David Willey and David Fletcher
talk about the development, production and action of the this tank. Check out the Tank Talk
about the T-26 to hear David Fletcher explain some more about the T-26 on The Tank Museum
YouTube Channel',
    2,
```

```
    '1940-01-28'
  );
INSERT INTO [Videos] (VideoUrl, VideoName, VideoDescription, VideoTypeId, VideoHistoricDate)
VALUES
  (
    'https://youtu.be/LSQUkgK2XP8',
    'Rommel, German Press and Polish Resistance - WW2 - OOTF 001',
    'This is the very first episode of Out of the Foxholes, in which we answer questions from
the community. In this first edition, we''ll talk about the early-war career of Erwin Rommel,
German Press on the Invasion of Poland and the birth of Polish resistance movements after the
German occupation in 1939',
    3,
    '1940-02-28'
  );
INSERT INTO [Videos] (VideoUrl, VideoName, VideoDescription, VideoTypeId, VideoHistoricDate)
VALUES
  (
    'https://youtu.be/gd5YhhNcC44',
    'Outbreak of the War Against Humanity - WW2 - War Against Humanity 001 - 5 March 1940',
    'When the Second World War breaks out, it is at first largely a war between one side of
totalitarian aggressors against a portion of the democratic countries of the world defending
other totalitarian states. From the first day of the war in Poland, as it already is in
China, this will be a war against humanity',
    1,
    '1940-03-05'
  );
GO

-- Add and fill out a couple playlists
INSERT INTO [Playlists] (PlaylistName, PlaylistDescription) VALUES ('All Videos', 'Every
single video, in real-time order');
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (0, 0);
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (0, 1);
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (0, 2);
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (0, 3);
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (0, 4);
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (0, 5);
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (0, 6);
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (0, 7);
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (0, 8);
INSERT INTO [Playlists] (PlaylistName, PlaylistDescription) VALUES ('The Annihilation of
Poland', 'The short-lived but valiant resistance of the Poles against Nazi Germany AND Soviet
Russia');
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (1, 0);
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (1, 1);
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (1, 2);
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (1, 3);
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (1, 4);
INSERT INTO [PlaylistVideos] (PlaylistId, VideoId) VALUES (1, 5);
GO
```

```sql
-- Add a couple of users. In reality the UserUnsubscribeKey will be automatically generated
by a trigger whenever a user is added
INSERT INTO [Users] (UserEmail, UserAlias, UserUnsubscribeKey) VALUES
  ('thejoshuaevans@gmail.com', 'Joshe Personal Email', 'exampleUnsubscribeKey0');
INSERT INTO [Users] (UserEmail, UserAlias, UserUnsubscribeKey) VALUES
  ('evansjj@my.lanecc.edu', 'Joshe School Email', 'exampleUnsubscribeKey1');
GO

-- Add the video user types. School joshe doesn't want to get WAH videos
INSERT INTO [UserVideoTypes] (UserId, VideoTypeId) VALUES (0, 0);
INSERT INTO [UserVideoTypes] (UserId, VideoTypeId) VALUES (0, 1);
INSERT INTO [UserVideoTypes] (UserId, VideoTypeId) VALUES (0, 2);
INSERT INTO [UserVideoTypes] (UserId, VideoTypeId) VALUES (0, 3);
INSERT INTO [UserVideoTypes] (UserId, VideoTypeId) VALUES (1, 0);
INSERT INTO [UserVideoTypes] (UserId, VideoTypeId) VALUES (1, 2);
INSERT INTO [UserVideoTypes] (UserId, VideoTypeId) VALUES (1, 3);
GO

-- Add the user playlists. School joshe is only interested in the initial Polish military
resistance
INSERT INTO [UserPlaylists] (UserId, PlaylistId, UserPlaylistStartYear) VALUES (0, 0, 2024);
INSERT INTO [UserPlaylists] (UserId, PlaylistId, UserPlaylistStartYear) VALUES (1, 1, 2024);
GO

-- Add some example emails. In reality these will be automatically generated every day
INSERT INTO [Emails] (UserId, EmailHistoricDate, EmailDate, EmailSubject, EmailBody) VALUES
  (0, '1939-09-01', '2024-09-01', 'What''s happening in WW2 on September 1st, 1939', '<BODY
FORMAT TO BE DETERMINED>')
INSERT INTO [Emails] (UserId, EmailHistoricDate, EmailDate, EmailSubject, EmailBody) VALUES
  (1, '1939-09-01', '2024-09-01', 'What''s happening in WW2 on September 1st, 1939', '<BODY
FORMAT TO BE DETERMINED>')
GO
```

Part 2 – Functions, Triggers, and Procedures

Functions

`GetHistoricDate`

Takes @CurrentDate (DATE) and @StartYear (INT) and generates a historic date using the following formula to generate the year (Note: @CurrentYear is the integer year segment of @CurrentDate). If the current date is a leap day and the historical date is not, the previous day will be returned instead

1939 – (@CurrentYear - @StartYear)

`GenerateEmailText`

Takes numerous inputs including the historical date, user information, and video information, and generates the HTML body of an email to send to the user. Likely one of the more complex functions, exact details are still under development, and may be broken into more functions still

Triggers

`BlockUserEmail`

Triggers whenever the [Users] table is updated or inserted into. If the new UserEmail is among the blocked users, will trigger an error

`CalculateUserUnsubscribeKey`

Triggers whenever the [Users] table is updated or inserted into. Will automatically calculate the UserUnsubscribeKey value for the user using the secret in the config table. Will overwrite any existing values

`ValidateUserEmail`

Triggers whenever the [Users] table is updated or inserted into and throws an error if the new UserEmail is not valid email address

Procedures

`GenerateDailyEmails`

For each user, and then for each of that user's playlist start years, generate email text using the GenerateEmailText function and use the values to fill in the [Emails] table. Ideally, would be automatically ran using a job at the beginning of every day

`InsertVideo`

Procedure used to insert videos. Takes all non-identity properties of the [Videos] table and uses them to perform a safe insertion into the table. Additionally, will create a new row in the [HistoricDates] table if needed

`InsertUser`

Procedure used to insert users. Takes all non-identity properties of the [Users] table, as well as a list of video type IDs so the [UserVideoTypes] can be automatically filled in

`UpdateUserVideoTypes`

Takes a @UserId (INT) and a comma delimited list of video IDs @VideoTypeIds (VARCHAR) and updates the user's video types to match the input values

`UpdateUserPlaylists`

Takes a @UserId (INT) and a comma delimited list of playlist IDs @PlaylistIds (VARCHAR) and updates the user's playlists to match the input values

`UpdatePlaylistVideosWithDateRange`

Takes @PlaylistId (INT), @StartDate (DATE), and @EndDate (DATE) and makes the identified playlist include all videos in the provided date range