

Advanced Database Systems

CE00332-6

Harry Clewlow (c012952a)

**Contents**

**1.0.0 SQL Design Pages 3 – 6**

1.1.0 Physical schema using a UML data modelling profile

**1.2.0** **SQL Implementation** **Pages 7 – 25**

1.2.0 Implementation using SQL Server with Functions Used Later

**1.3.0 SQL Queries Pages 26 – 29**

1.3.1 Your Choice Show Query

1.3.2 Beaconside Community Choir Query

1.3.3 Staff Shows, Songs and Employed Status Query

1.3.4 Comment Length, Status for Weekly Shows Query

**1.4.0** **SQL Object Elements** **Pages 29 – 34**

1.4.1 Number of Times Played Trigger

1.4.2 Populating Playlist with Requests / Dedications Trigger

1.4.2 Most Request Songs Not In the Database Procedure

1.4.2 Backup Popular Songs Played More than X Times Procedure

**2.1.0 MongoDB Design Pages 34 – 35**

2.1.0 Pseudo Code for the Customer Comments Log

**2.2.0 MongoDB Implementation Pages 35 – 36**

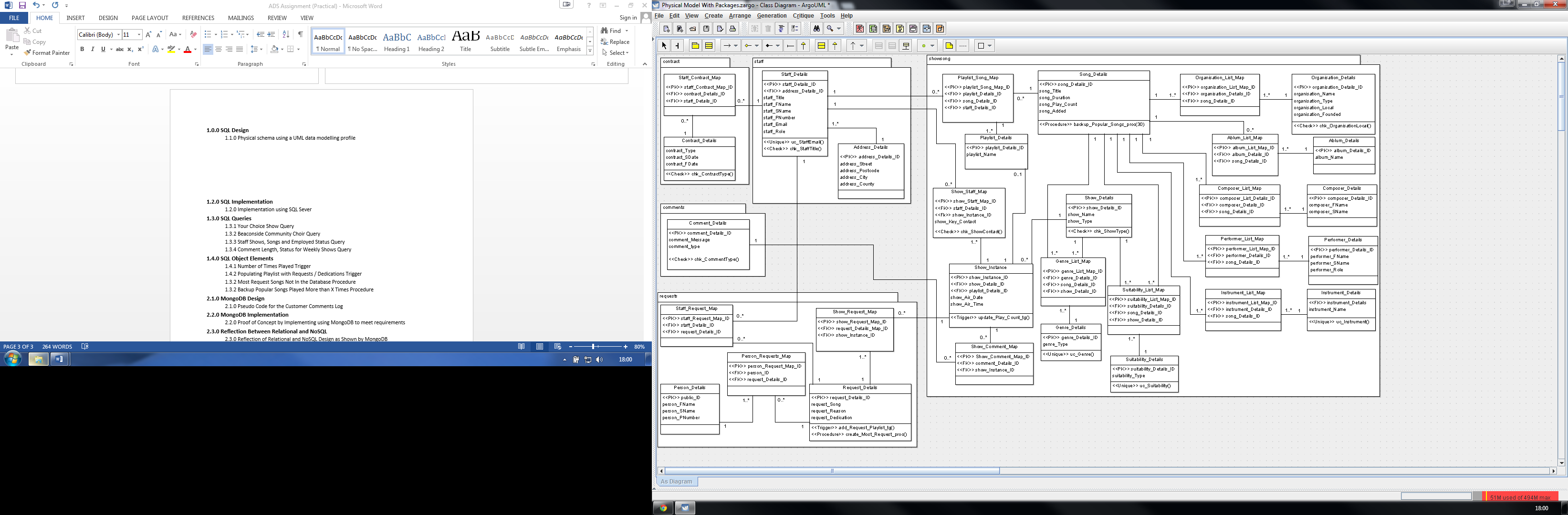
2.2.0 Proof of Concept by Implementing using MongoDB to meet requirements

**2.3.0 Reflection Between Relational and NoSQL Pages 36 - 37**

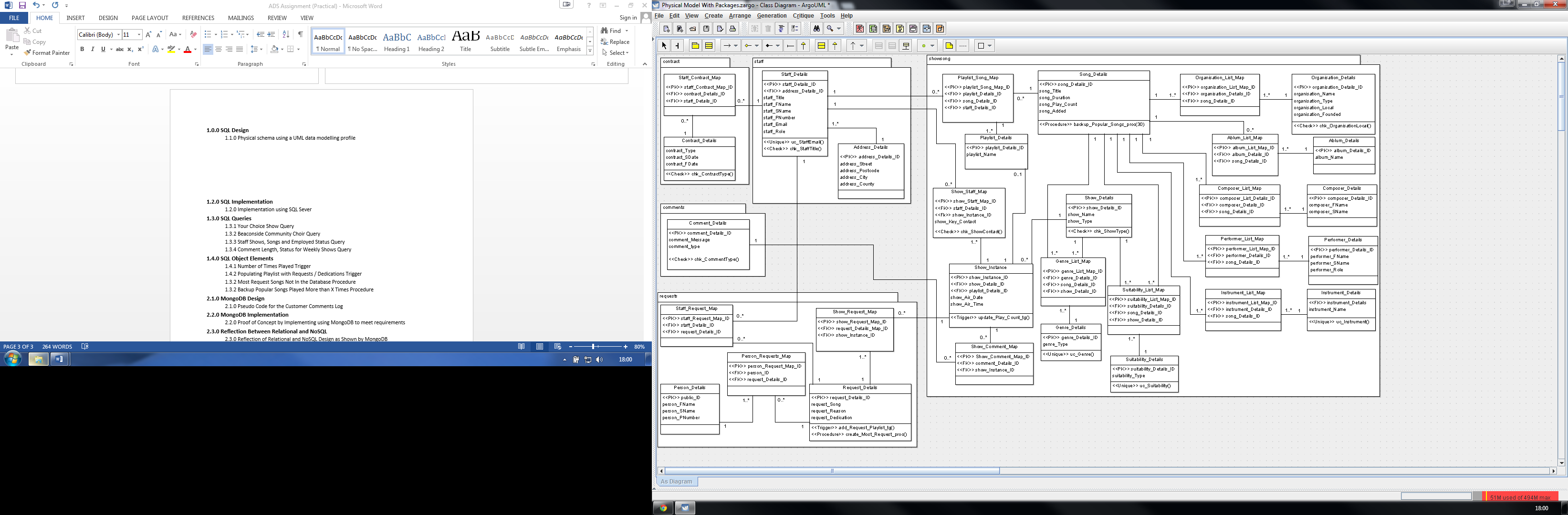
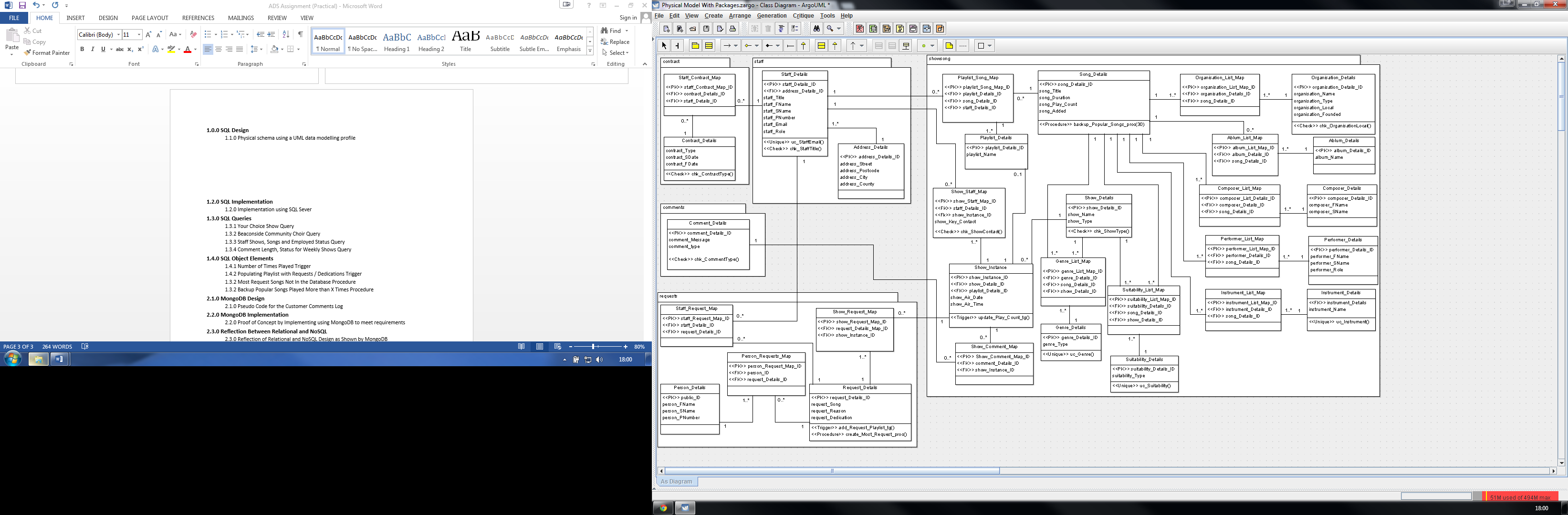
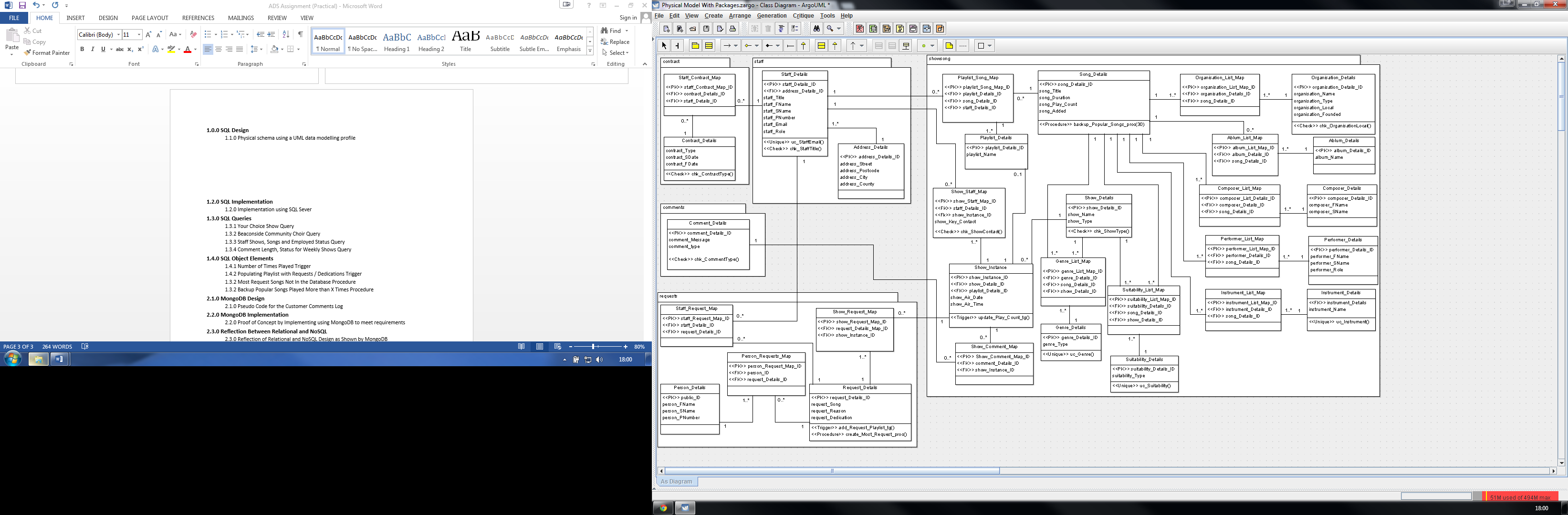
2.3.0 Reflection of Relational and NoSQL Design as Shown by MongoDB

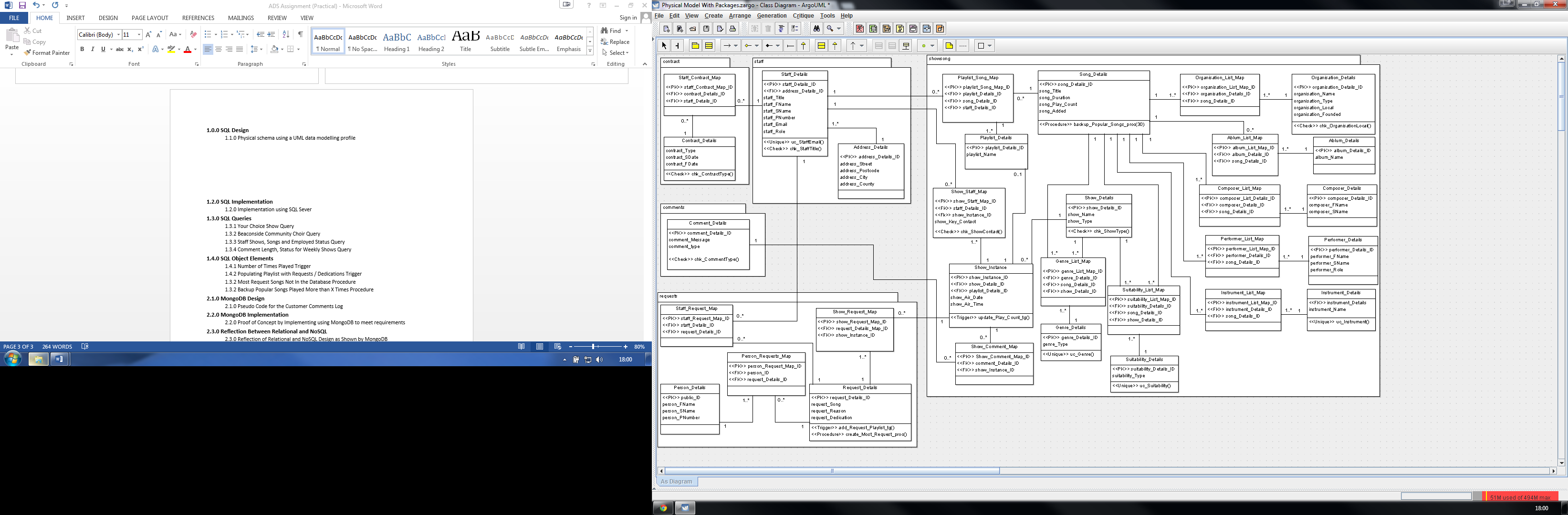
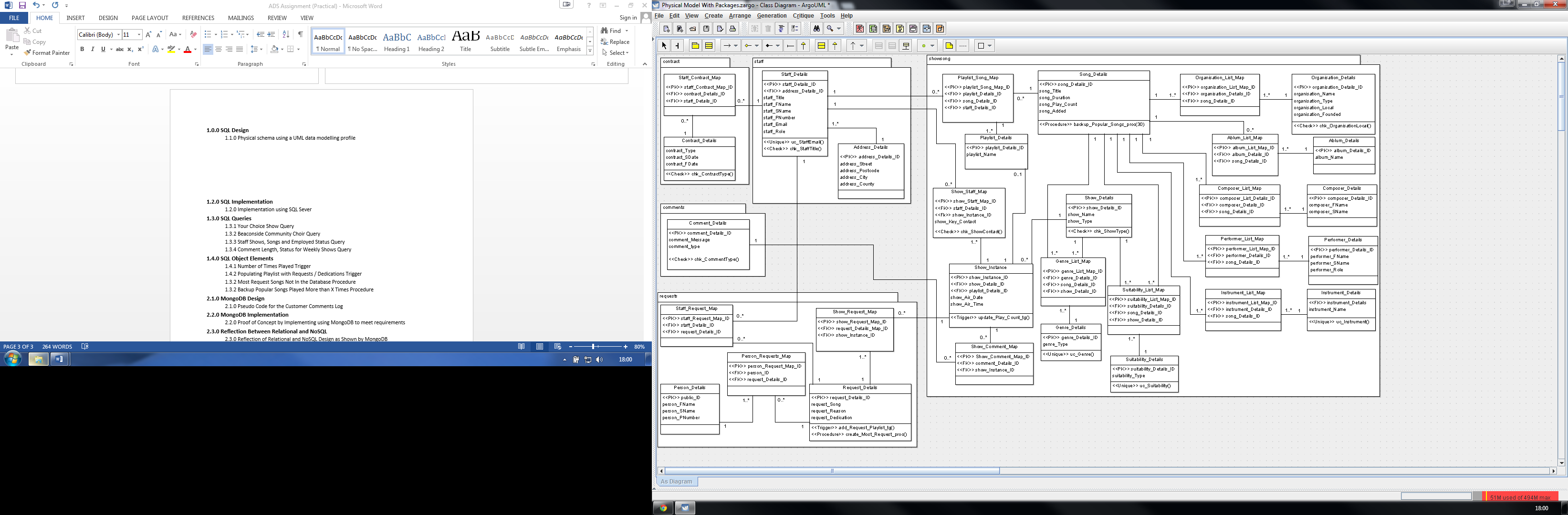
**1.0.0 SQL Design**

1.1.0 Physical schema using a UML data modelling profile



Below I have separated the packages so that you can see the tables, columns, fields, check constraints and other object elements due to the full UML being very small in the above image.





**1.2.0** **SQL Implementation**

1.2.0 Implementation using SQL Server with Functions Used Later

CREATE TABLE Address\_Details

(

address\_Details\_ID int IDENTITY(1,1) NOT NULL,

address\_Street varchar (35) NOT NULL,

address\_Postcode varchar (9) NOT NULL,

address\_City varchar (20) NOT NULL,

address\_County varchar (20) NOT NULL,

CONSTRAINT pk\_AddressID PRIMARY KEY (address\_Details\_ID)

);

CREATE TABLE Staff\_Details

(

staff\_Details\_ID int IDENTITY(1,1) NOT NULL,

address\_Details\_ID int NOT NULL,

staff\_Title varchar (6),

staff\_FName varchar (35) NOT NULL,

staff\_SName varchar (35) NOT NULL,

staff\_PNumber varchar (11),

staff\_Email varchar (255),

staff\_Role varchar (50) NOT NULL,

CONSTRAINT pk\_StaffID PRIMARY KEY (staff\_Details\_ID),

CONSTRAINT fk\_AddressID FOREIGN KEY (address\_Details\_ID) REFERENCES Address\_Details (address\_Details\_ID),

CONSTRAINT uc\_StaffEmail UNIQUE (staff\_Email),

CONSTRAINT chk\_StaffTitle CHECK (staff\_Title ='Mr'OR staff\_Title = 'Mrs' OR staff\_Title = 'Ms' OR staff\_Title = 'Master' OR staff\_Title = 'Miss')

);

CREATE TABLE Contract\_Details

(

contract\_Details\_ID int IDENTITY(1,1) NOT NULL,

contract\_Type varchar (35) NOT NULL,

contract\_SDate date NOT NULL,

contract\_FDate date,

CONSTRAINT pk\_ContractID PRIMARY KEY (contract\_Details\_ID),

CONSTRAINT chk\_ContractType CHECK (contract\_Type ='Full Time Permanent' OR contract\_Type ='Part Time' OR contract\_Type ='Casual Contract' OR contract\_Type ='Volunteer' OR contract\_Type ='Intern')

);

CREATE TABLE Staff\_Contract\_Map

(

staff\_Contract\_Map\_ID int IDENTITY(1,1) NOT NULL,

contract\_Details\_ID int NOT NULL,

staff\_Details\_ID int NOT NULL,

CONSTRAINT pk\_StaffContractID PRIMARY KEY (staff\_Contract\_Map\_ID),

CONSTRAINT fk\_ContractID FOREIGN KEY (contract\_Details\_ID) REFERENCES Contract\_Details (contract\_Details\_ID),

CONSTRAINT fk\_StaffID\_SCMap FOREIGN KEY (staff\_Details\_ID) REFERENCES Staff\_Details (staff\_Details\_ID),

);

CREATE TABLE Comment\_Details

(

comment\_Details\_ID int IDENTITY(1,1) NOT NULL,

comment\_Message varchar (255) NOT NULL,

comment\_Type varchar (8) NOT NULL,

CONSTRAINT pk\_CommentID PRIMARY KEY (comment\_Details\_ID),

CONSTRAINT chk\_CommentType CHECK (comment\_Type ='Positive' OR comment\_Type ='Negative')

);

CREATE TABLE Suitability\_Details

(

suitability\_Details\_ID int IDENTITY(1,1) NOT NULL,

suitability\_Type varchar (16) NOT NULL,

CONSTRAINT pk\_SuitabilityID PRIMARY KEY (suitability\_Details\_ID),

CONSTRAINT uc\_Suitability UNIQUE (suitability\_Type)

);

CREATE TABLE Instrument\_Details

(

instrument\_Details\_ID int IDENTITY(1,1) NOT NULL,

instrument\_Name varchar (50) NOT NULL,

CONSTRAINT pk\_InstrumentID PRIMARY KEY (instrument\_Details\_ID)

);

CREATE TABLE Genre\_Details

(

genre\_Details\_ID int IDENTITY(1,1) NOT NULL,

genre\_Type varchar (40) NOT NULL,

CONSTRAINT pk\_GenreID PRIMARY KEY (genre\_Details\_ID),

CONSTRAINT uc\_Genre UNIQUE (genre\_Type),

);

CREATE TABLE Show\_Details

(

show\_Details\_ID int IDENTITY(1,1) NOT NULL,

show\_Name varchar (255) NOT NULL,

show\_Type varchar (20) NOT NULL,

CONSTRAINT pk\_ShowID PRIMARY KEY (show\_Details\_ID),

CONSTRAINT chk\_ShowType CHECK (show\_Type ='One Off Broadcast' OR show\_Type ='Daily' OR show\_Type ='Weekly' OR show\_Type ='Monthly')

);

CREATE TABLE Song\_Details

(

song\_Details\_ID int IDENTITY(1,1) NOT NULL,

song\_Title varchar (255) NOT NULL,

song\_Duration time NOT NULL,

song\_Play\_Count int DEFAULT 0,

song\_Added date DEFAULT getDate(),

CONSTRAINT pk\_SongID PRIMARY KEY (song\_Details\_ID)

);

CREATE TABLE Organisation\_Details

(

organisation\_Details\_ID int IDENTITY(1,1) NOT NULL,

organisation\_Name varchar (255) NOT NULL,

organisation\_Type varchar (255) NOT NULL,

organisation\_Local bit NOT NULL,

organisation\_Founded date,

CONSTRAINT pk\_OrganisationID PRIMARY KEY (organisation\_Details\_ID),

CONSTRAINT chk\_OrganisationLocal CHECK (organisation\_Local = 0 OR organisation\_Local = 1)

);

CREATE TABLE Album\_Details

(

album\_Details\_ID int IDENTITY(1,1) NOT NULL,

album\_Name varchar (255) NOT NULL,

CONSTRAINT pk\_AlbumID PRIMARY KEY (album\_Details\_ID)

);

CREATE TABLE Composer\_Details

(

composer\_Details\_ID int IDENTITY(1,1) NOT NULL,

composer\_FName varchar (35) NOT NULL,

composer\_SName varchar (35) NOT NULL,

CONSTRAINT pk\_ComposerID PRIMARY KEY (composer\_Details\_ID)

);

CREATE TABLE Performer\_Details

(

performer\_Details\_ID int IDENTITY(1,1) NOT NULL,

performer\_FName varchar (35) NOT NULL,

performer\_SName varchar (35) NOT NULL,

performer\_Role varchar (50) NOT NULL,

CONSTRAINT pk\_PerformerID PRIMARY KEY (performer\_Details\_ID)

);

CREATE TABLE Suitability\_List\_Map

(

suitability\_List\_Map\_ID int IDENTITY(1,1) NOT NULL,

suitability\_Details\_ID int NOT NULL,

song\_Details\_ID int,

show\_Details\_ID int,

CONSTRAINT pk\_SuitabilityListID PRIMARY KEY (suitability\_List\_Map\_ID),

CONSTRAINT fk\_SuitabilityID FOREIGN KEY (suitability\_Details\_ID) REFERENCES Suitability\_Details (suitability\_Details\_ID),

CONSTRAINT fk\_SongID\_Suitability FOREIGN KEY (song\_Details\_ID) REFERENCES Song\_Details (song\_Details\_ID),

CONSTRAINT fk\_ShowID\_Suitability FOREIGN KEY (show\_Details\_ID) REFERENCES Show\_Details (show\_Details\_ID)

);

CREATE TABLE Composer\_List\_Map

(

composer\_List\_Map\_ID int IDENTITY(1,1) NOT NULL,

composer\_Details\_ID int NOT NULL,

song\_Details\_ID int NOT NULL,

CONSTRAINT pk\_ComposerListID PRIMARY KEY (composer\_List\_Map\_ID),

CONSTRAINT fk\_ComposerID FOREIGN KEY (composer\_Details\_ID) REFERENCES Composer\_Details (composer\_Details\_ID),

CONSTRAINT fk\_SongID\_Composer FOREIGN KEY (song\_Details\_ID) REFERENCES Song\_Details (song\_Details\_ID)

);

CREATE TABLE Performer\_List\_Map

(

performer\_List\_Map\_ID int IDENTITY(1,1) NOT NULL,

performer\_Details\_ID int NOT NULL,

song\_Details\_ID int NOT NULL,

CONSTRAINT pk\_PerformerListID PRIMARY KEY (performer\_List\_Map\_ID),

CONSTRAINT fk\_PerformerID FOREIGN KEY (performer\_Details\_ID) REFERENCES Performer\_Details (performer\_Details\_ID),

CONSTRAINT fk\_SongID\_Performer FOREIGN KEY (song\_Details\_ID) REFERENCES Song\_Details (song\_Details\_ID)

);

CREATE TABLE Album\_List\_Map

(

album\_List\_Map\_ID int IDENTITY(1,1) NOT NULL,

album\_Details\_ID int NOT NULL,

song\_Details\_ID int NOT NULL,

CONSTRAINT pk\_AlbumListID PRIMARY KEY (album\_List\_Map\_ID),

CONSTRAINT fk\_AlbumID FOREIGN KEY (album\_Details\_ID) REFERENCES Album\_Details (album\_Details\_ID),

CONSTRAINT fk\_SongID\_Album FOREIGN KEY (song\_Details\_ID) REFERENCES Song\_Details (song\_Details\_ID)

);

CREATE TABLE Organisation\_List\_Map

(

organisation\_List\_Map int IDENTITY(1,1) NOT NULL,

organisation\_Details\_ID int NOT NULL,

song\_Details\_ID int NOT NULL,

CONSTRAINT pk\_OrganisationListID PRIMARY KEY (organisation\_List\_Map),

CONSTRAINT fk\_OrganisationID FOREIGN KEY (organisation\_Details\_ID) REFERENCES Organisation\_Details (organisation\_Details\_ID),

CONSTRAINT fk\_SongID\_Organisation FOREIGN KEY (song\_Details\_ID) REFERENCES Song\_Details (song\_Details\_ID)

);

CREATE TABLE Instrument\_List\_Map

(

instrument\_List\_Map int IDENTITY(1,1) NOT NULL,

instrument\_Details\_ID int NOT NULL,

song\_Details\_ID int NOT NULL,

CONSTRAINT pk\_InstrumentListID PRIMARY KEY (instrument\_List\_Map),

CONSTRAINT fk\_InstrumentID FOREIGN KEY (instrument\_Details\_ID) REFERENCES Instrument\_Details (instrument\_Details\_ID),

CONSTRAINT fk\_SongID\_Instrument FOREIGN KEY (song\_Details\_ID) REFERENCES Song\_Details (song\_Details\_ID)

);

CREATE TABLE Genre\_List\_Map

(

genre\_List\_Map int IDENTITY(1,1) NOT NULL,

genre\_Details\_ID int NOT NULL,

song\_Details\_ID int,

show\_Details\_ID int,

CONSTRAINT pk\_GenreListID PRIMARY KEY (genre\_List\_Map),

CONSTRAINT fk\_GenreID FOREIGN KEY (genre\_Details\_ID) REFERENCES Genre\_Details (genre\_Details\_ID),

CONSTRAINT fk\_SongID\_Genre FOREIGN KEY (song\_Details\_ID) REFERENCES Song\_Details (song\_Details\_ID),

CONSTRAINT fk\_ShowID\_Genre FOREIGN KEY (show\_Details\_ID) REFERENCES Show\_Details (show\_Details\_ID)

);

CREATE TABLE Playlist\_Details

(

playlist\_Details\_ID int IDENTITY(1,1) NOT NULL,

playlist\_Name varchar (255) NOT NULL,

CONSTRAINT pk\_PlaylistID PRIMARY KEY (playlist\_Details\_ID)

);

CREATE TABLE Show\_Instance

(

show\_Instance\_ID int IDENTITY(1,1) NOT NULL,

show\_Details\_ID int NOT NULL,

playlist\_Details\_ID int,

show\_Air\_Date date NOT NULL,

show\_Air\_Time time NOT NULL,

CONSTRAINT pk\_ShowInstanceID PRIMARY KEY (show\_Instance\_ID),

CONSTRAINT fk\_ShowID\_ShowInstance FOREIGN KEY (show\_Details\_ID) REFERENCES Show\_Details (show\_Details\_ID),

CONSTRAINT fk\_PlaylistID\_ShowInstance FOREIGN KEY (playlist\_Details\_ID) REFERENCES Playlist\_Details (playlist\_Details\_ID)

);

CREATE TABLE Show\_Comment\_Map

(

Show\_Comment\_Map\_ID int IDENTITY(1,1) NOT NULL,

comment\_Details\_ID int NOT NULL,

show\_Instance\_ID int NOT NULL,

CONSTRAINT pk\_ShowCommentID PRIMARY KEY (Show\_Comment\_Map\_ID),

CONSTRAINT fk\_CommentID FOREIGN KEY (comment\_Details\_ID) REFERENCES Comment\_Details (comment\_Details\_ID),

CONSTRAINT fk\_ShowInstanceID\_SCMap FOREIGN KEY (show\_Instance\_ID) REFERENCES Show\_Instance (show\_Instance\_ID)

);

CREATE TABLE Show\_Staff\_Map

(

show\_Staff\_Map int IDENTITY(1,1) NOT NULL,

staff\_Details\_ID int NOT NULL,

show\_Instance\_ID int NOT NULL,

show\_Key\_Contact bit NOT NULL,

CONSTRAINT pk\_ShowStaffID PRIMARY KEY (show\_Staff\_Map),

CONSTRAINT fk\_Staff\_Details\_SSMap FOREIGN KEY (staff\_Details\_ID) REFERENCES Staff\_Details (staff\_Details\_ID),

CONSTRAINT fk\_ShowInstanceID\_SSMap FOREIGN KEY (show\_Instance\_ID) REFERENCES Show\_Instance (show\_Instance\_ID),

CONSTRAINT chk\_ShowContact CHECK (show\_Key\_Contact = 0 OR show\_Key\_Contact = 1)

);

CREATE TABLE Playlist\_Song\_Map

(

playlist\_Song\_Map\_ID int IDENTITY(1,1) NOT NULL,

playlist\_Details\_ID int NOT NULL,

song\_Details\_ID int NOT NULL,

staff\_Details\_ID int NOT NULL,

CONSTRAINT pk\_PlaylistSongID PRIMARY KEY (playlist\_Song\_Map\_ID),

CONSTRAINT fk\_PlaylistID FOREIGN KEY (playlist\_Details\_ID) REFERENCES Playlist\_Details (playlist\_Details\_ID),

CONSTRAINT fk\_SongID\_PSMap FOREIGN KEY (song\_Details\_ID) REFERENCES Song\_Details (song\_Details\_ID),

CONSTRAINT fk\_ShowID\_PSMap FOREIGN KEY (staff\_Details\_ID) REFERENCES Staff\_Details (staff\_Details\_ID)

);

CREATE TABLE Person\_Details

(

person\_Details\_ID int IDENTITY(1,1) NOT NULL,

person\_FName varchar (35) NOT NULL,

person\_SName varchar (35) NOT NULL,

person\_PNumber varchar (11) NOT NULL,

CONSTRAINT pk\_personID PRIMARY KEY (person\_Details\_ID)

);

CREATE TABLE Request\_Details

(

request\_Details\_ID int IDENTITY(1,1) NOT NULL,

request\_Song varchar (255) NOT NULL,

request\_Reason varchar (255) NOT NULL,

request\_Dedication varchar (255),

CONSTRAINT pk\_RequestID PRIMARY KEY (request\_Details\_ID)

);

CREATE TABLE Person\_Request\_Map

(

person\_Request\_Map int IDENTITY(1,1) NOT NULL,

person\_Details\_ID int NOT NULL,

request\_Details\_ID int NOT NULL

CONSTRAINT pk\_PersonReuqestID PRIMARY KEY (person\_Request\_Map),

CONSTRAINT fk\_PersonID FOREIGN KEY (person\_Details\_ID) REFERENCES Person\_Details (person\_Details\_ID),

CONSTRAINT fk\_RequestID\_Public FOREIGN KEY (request\_Details\_ID) REFERENCES Request\_Details (request\_Details\_ID)

);

CREATE TABLE Staff\_Request\_Map

(

staff\_Request\_Map int IDENTITY(1,1) NOT NULL,

staff\_Details\_ID int NOT NULL,

request\_Details\_ID int NOT NULL

CONSTRAINT pk\_StaffReuqestID PRIMARY KEY (staff\_Request\_Map),

CONSTRAINT fk\_StaffID\_Request FOREIGN KEY (staff\_Details\_ID) REFERENCES Staff\_Details (staff\_Details\_ID),

CONSTRAINT fk\_RequestID\_Staff FOREIGN KEY (request\_Details\_ID) REFERENCES Request\_Details (request\_Details\_ID)

);

CREATE TABLE Show\_Request\_Map

(

show\_Request\_Map int IDENTITY(1,1) NOT NULL,

request\_Details\_ID int NOT NULL,

show\_Instance\_ID int NOT NULL,

CONSTRAINT pk\_ShowReuqestID PRIMARY KEY (show\_Request\_Map),

CONSTRAINT fk\_ShowInstanceID\_Request FOREIGN KEY (show\_Instance\_ID) REFERENCES Show\_Instance (show\_Instance\_ID),

CONSTRAINT fk\_RequestID\_Show FOREIGN KEY (request\_Details\_ID) REFERENCES Request\_Details (request\_Details\_ID)

);

CREATE NONCLUSTERED INDEX nc\_index\_SongTitle ON Song\_Details(song\_Title);

CREATE NONCLUSTERED INDEX nc\_index\_OrganisationLocal ON Organisation\_Details(organisation\_Local);

INSERT INTO Address\_Details VALUES ('122 Corporation Street', 'ST16 3LS', 'Stafford','Staffordshire');

INSERT INTO Address\_Details VALUES ('23 Meyrick Road', 'ST17 4DG', 'Stafford','Staffordshire');

INSERT INTO Address\_Details VALUES ('102 Corporation Street', 'ST16 1LS', 'Stafford','Staffordshire');

INSERT INTO Address\_Details VALUES ('Summer Hill Farm', 'ST13 2FH', 'Stoke','Staffordshire');

INSERT INTO Address\_Details VALUES ('Rosy Lane', 'SD1 31LS', 'Eccleshall','Staffordshire');

INSERT INTO Staff\_Details VALUES (1,'Mr','Harry','Clewlow','0777777777','Harry.Clewlow.1993@gmail.com','DJ, Producer');

INSERT INTO Staff\_Details VALUES (2,'Mrs','Sam','Smith','0773777777','Sam.Smith.1973@gmail.com',' Producer');

INSERT INTO Staff\_Details VALUES (3,'Master','Dave','Paulson','0771777777','Dave.Paulson.1997@yahoo.com','DJ');

INSERT INTO Staff\_Details VALUES (4,'Ms','Jess','Robinson','0727777777','Jess.Robinson.1963@me.com','Producer, Host');

INSERT INTO Staff\_Details VALUES (5,'Miss','Laura','Sutton','0777777747','Laura.Sutton.1993@hotmail.com','DJ, Host');

INSERT INTO Contract\_Details VALUES ('Intern', '20120202','20130423');

INSERT INTO Contract\_Details VALUES ('Full Time Permanent', '20120424', NULL);

INSERT INTO Contract\_Details VALUES ('Volunteer', '20110213', NULL);

INSERT INTO Contract\_Details VALUES ('Part Time', '20100513', NULL);

INSERT INTO Contract\_Details VALUES ('Casual Contract', '20120722', NULL);

INSERT INTO Contract\_Details VALUES ('Full Time Permanent', '20130606', NULL);

INSERT INTO Staff\_Contract\_Map VALUES (1, 1);

INSERT INTO Staff\_Contract\_Map VALUES (2, 1);

INSERT INTO Staff\_Contract\_Map VALUES (3, 2);

INSERT INTO Staff\_Contract\_Map VALUES (4, 3);

INSERT INTO Staff\_Contract\_Map VALUES (5, 4);

INSERT INTO Staff\_Contract\_Map VALUES (6, 5);

INSERT INTO Comment\_Details VALUES ('Nice Show ', 'Positive');

INSERT INTO Comment\_Details VALUES ('Not Your Best Show', 'Negative');

INSERT INTO Comment\_Details VALUES ('Really Bad Show Never Listening Again', 'Negative');

INSERT INTO Comment\_Details VALUES ('Amazing Show Guys !!!!', 'Positive');

INSERT INTO Comment\_Details VALUES ('This Show Needs More Air Time, Keep Up The Great Work', 'Positive');

INSERT INTO Comment\_Details VALUES ('Why Talk About Dogs For An Hour MORE MUSIC PLEASE!!', 'Negative');

INSERT INTO Comment\_Details VALUES ('Nice To See Special Guests On The Show', 'Positive');

INSERT INTO Comment\_Details VALUES ('WORST SHOW SO FAR !!!!!!!!', 'Negative');

INSERT INTO Comment\_Details VALUES ('Good Shows Today', 'Positive');

INSERT INTO Comment\_Details VALUES ('THIS WEEK HAD REALLY BAD MUSIC', 'Negative');

INSERT INTO Comment\_Details VALUES ('Nice', 'Positive');

INSERT INTO Comment\_Details VALUES ('The Adverts Are Not Relevant To Me', 'Negative');

INSERT INTO Suitability\_Details VALUES ('Pre-School');

INSERT INTO Suitability\_Details VALUES ('Junior School');

INSERT INTO Suitability\_Details VALUES ('Secondary School');

INSERT INTO Suitability\_Details VALUES ('Adult');

INSERT INTO Suitability\_Details VALUES ('Older Citizen');

INSERT INTO Suitability\_Details VALUES ('Risky');

INSERT INTO Instrument\_Details VALUES ('Piano');

INSERT INTO Instrument\_Details VALUES ('Triangle');

INSERT INTO Instrument\_Details VALUES ('Drums');

INSERT INTO Instrument\_Details VALUES ('Acoustic Guitar');

INSERT INTO Instrument\_Details VALUES ('Electric Guitar');

INSERT INTO Instrument\_Details VALUES ('Keyboard');

INSERT INTO Instrument\_Details VALUES ('Violin');

INSERT INTO Instrument\_Details VALUES ('Cello');

INSERT INTO Instrument\_Details VALUES ('Bass Guitar');

INSERT INTO Instrument\_Details VALUES ('Saxophone');

INSERT INTO Instrument\_Details VALUES ('Trumpet');

INSERT INTO Instrument\_Details VALUES ('Electronically Generated');

INSERT INTO Instrument\_Details VALUES ('Synthesizer');

INSERT INTO Genre\_Details VALUES ('Alternative');

INSERT INTO Genre\_Details VALUES ('Blues');

INSERT INTO Genre\_Details VALUES ('Classical');

INSERT INTO Genre\_Details VALUES ('Country');

INSERT INTO Genre\_Details VALUES ('Dance');

INSERT INTO Genre\_Details VALUES ('Easy Listening');

INSERT INTO Genre\_Details VALUES ('Electronic');

INSERT INTO Genre\_Details VALUES ('European');

INSERT INTO Genre\_Details VALUES ('Hip Hop / Rap');

INSERT INTO Genre\_Details VALUES ('Indie Pop');

INSERT INTO Genre\_Details VALUES ('Asian Pop');

INSERT INTO Genre\_Details VALUES ('Jazz');

INSERT INTO Genre\_Details VALUES ('Opera');

INSERT INTO Genre\_Details VALUES ('Musical');

INSERT INTO Genre\_Details VALUES ('Popular');

INSERT INTO Genre\_Details VALUES ('Modern');

INSERT INTO Show\_Details VALUES ('Your Choice Show','Daily');

INSERT INTO Show\_Details VALUES ('Sunday Night Live','Weekly');

INSERT INTO Show\_Details VALUES ('Not Your Choice','Daily');

INSERT INTO Show\_Details VALUES ('Chart Show','Weekly');

INSERT INTO Show\_Details VALUES ('Late Nights With Sam','One Off Broadcast');

INSERT INTO Show\_Details VALUES ('Breakfast Show','Daily');

INSERT INTO Show\_Details VALUES ('Classics Hour','Weekly');

INSERT INTO Show\_Details VALUES ('Sports Zone','Monthly');

INSERT INTO Show\_Details VALUES ('Requests Show','Weekly');

INSERT INTO Show\_Details VALUES ('Dedication Show','Weekly');

INSERT INTO Performer\_Details VALUES ('Paul', 'Smith', 'Lead Vocals');

INSERT INTO Performer\_Details VALUES ('Dan', 'Smath', 'Lead Guitar');

INSERT INTO Performer\_Details VALUES ('Alex', 'Boxman', 'Singer');

INSERT INTO Performer\_Details VALUES ('Dan', 'Cueton', 'Pianist');

INSERT INTO Performer\_Details VALUES ('Patrick', 'Croft', 'Guitar');

INSERT INTO Performer\_Details VALUES ('Dave', 'Yau', 'Bass');

INSERT INTO Performer\_Details VALUES ('Sam', 'Burton', 'Pianist ');

INSERT INTO Performer\_Details VALUES ('Harry', 'Cambridge', 'DJ');

INSERT INTO Performer\_Details VALUES ('Paul', 'Stafford', 'Singer');

INSERT INTO Performer\_Details VALUES ('Bill', 'Cave', 'Vocals');

INSERT INTO Performer\_Details VALUES ('Ronnie', 'Green', 'Violin');

INSERT INTO Performer\_Details VALUES ('George', 'Blue', 'Pianist');

INSERT INTO Album\_Details VALUES ('The Greatest Hit');

INSERT INTO Album\_Details VALUES ('Really Bad Songs');

INSERT INTO Album\_Details VALUES ('WOW You Listen To This');

INSERT INTO Organisation\_Details VALUES ('The Red Stones', 'Band', 1, '20120501');

INSERT INTO Organisation\_Details VALUES ('Ellesmere Junior Choir', 'Choir', 0, '20121107');

INSERT INTO Organisation\_Details VALUES ('Ellesmere Senior Orchestra', 'School Orchestra', 1, '20120209');

INSERT INTO Organisation\_Details VALUES ('The Bloody Wolves', 'Band', 0, '20131215');

INSERT INTO Organisation\_Details VALUES ('Beaconside Community Choir', 'Choir', 1, '20131215');

INSERT INTO Composer\_Details VALUES ('Paul', 'Stafford');

INSERT INTO Composer\_Details VALUES ('Bill', 'Cave');

INSERT INTO Composer\_Details VALUES ('Ronnie', 'Green');

INSERT INTO Composer\_Details VALUES ('George', 'Blue');

INSERT INTO Song\_Details VALUES ('A Song', '00:02:00', default, '20101209');

INSERT INTO Song\_Details VALUES ('B Song', '00:15:30', default, '20100906');

INSERT INTO Song\_Details VALUES ('C Song', '00:45:30', default, '20100104');

INSERT INTO Song\_Details VALUES ('D Song', '00:03:32', default, '20100203');

INSERT INTO Song\_Details VALUES ('E Song', '00:03:12', default, '20101202');

INSERT INTO Song\_Details VALUES ('F Song', '00:04:37', default, '20101106');

INSERT INTO Song\_Details VALUES ('G Song', '00:05:56', default, '20100701');

INSERT INTO Song\_Details VALUES ('H Song', '00:05:30', default, '20100308');

INSERT INTO Song\_Details VALUES ('I Song', '00:02:30', default, '20100210');

INSERT INTO Song\_Details VALUES ('J Song', '00:02:30', default, '20100511');

INSERT INTO Song\_Details VALUES ('K Song', '00:02:30', default, default);

INSERT INTO Playlist\_Details VALUES ('Your Choice Show #1');

INSERT INTO Playlist\_Details VALUES ('Sunday Night Live #12');

INSERT INTO Playlist\_Details VALUES ('Not Your Choice Show #3');

INSERT INTO Playlist\_Details VALUES ('Chart Show #43');

INSERT INTO Playlist\_Details VALUES ('Breakfast Show #653');

INSERT INTO Playlist\_Details VALUES ('Late Nights With Sam # 9');

INSERT INTO Playlist\_Details VALUES ('Sports Zone #54');

INSERT INTO Playlist\_Details VALUES ('Classics Hour #254');

INSERT INTO Playlist\_Details VALUES ('Requests Show #54');

INSERT INTO Playlist\_Details VALUES ('Dedication Show #1');

INSERT INTO Playlist\_Song\_Map VALUES (1, 1, 1);

INSERT INTO Playlist\_Song\_Map VALUES (1, 2, 1);

INSERT INTO Playlist\_Song\_Map VALUES (1, 3, 1);

INSERT INTO Playlist\_Song\_Map VALUES (2, 4, 4);

INSERT INTO Playlist\_Song\_Map VALUES (2, 5, 4);

INSERT INTO Playlist\_Song\_Map VALUES (2, 6, 4);

INSERT INTO Playlist\_Song\_Map VALUES (3, 7, 1);

INSERT INTO Playlist\_Song\_Map VALUES (3, 8, 1);

INSERT INTO Playlist\_Song\_Map VALUES (3, 9, 1);

INSERT INTO Playlist\_Song\_Map VALUES (4, 10, 5);

INSERT INTO Playlist\_Song\_Map VALUES (4, 9, 5);

INSERT INTO Playlist\_Song\_Map VALUES (4, 8, 5);

INSERT INTO Playlist\_Song\_Map VALUES (5, 7, 2);

INSERT INTO Playlist\_Song\_Map VALUES (5, 6, 2);

INSERT INTO Playlist\_Song\_Map VALUES (5, 5, 2);

INSERT INTO Playlist\_Song\_Map VALUES (6, 4, 2);

INSERT INTO Playlist\_Song\_Map VALUES (6, 3, 2);

INSERT INTO Playlist\_Song\_Map VALUES (6, 2, 2);

INSERT INTO Playlist\_Song\_Map VALUES (7, 1, 1);

INSERT INTO Playlist\_Song\_Map VALUES (7, 2, 1);

INSERT INTO Playlist\_Song\_Map VALUES (7, 3, 1);

INSERT INTO Playlist\_Song\_Map VALUES (8, 4, 5);

INSERT INTO Playlist\_Song\_Map VALUES (8, 5, 5);

INSERT INTO Playlist\_Song\_Map VALUES (8, 6, 5);

INSERT INTO Genre\_List\_Map VALUES (1, 1, 1);

INSERT INTO Genre\_List\_Map VALUES (2, 2, 5);

INSERT INTO Genre\_List\_Map VALUES (3, 4, 1);

INSERT INTO Genre\_List\_Map VALUES (4, 1, 4);

INSERT INTO Genre\_List\_Map VALUES (5, 2, 1);

INSERT INTO Genre\_List\_Map VALUES (6, 1, 3);

INSERT INTO Genre\_List\_Map VALUES (7, 7, NULL);

INSERT INTO Genre\_List\_Map VALUES (8, 9, 6);

INSERT INTO Genre\_List\_Map VALUES (9, 1, NULL);

INSERT INTO Genre\_List\_Map VALUES (10,10, NULL);

INSERT INTO Genre\_List\_Map VALUES (11, 3, 1);

INSERT INTO Genre\_List\_Map VALUES (12, NULL, 1);

INSERT INTO Genre\_List\_Map VALUES (13, 1, 3);

INSERT INTO Genre\_List\_Map VALUES (14, 5, NULL);

INSERT INTO Genre\_List\_Map VALUES (15, 7, 2);

INSERT INTO Genre\_List\_Map VALUES (16, NULL, 7);

INSERT INTO Genre\_List\_Map VALUES (2, 6, 8);

INSERT INTO Genre\_List\_Map VALUES (3, 8, 9);

INSERT INTO Suitability\_List\_Map VALUES (1, 1, 1);

INSERT INTO Suitability\_List\_Map VALUES (2, 1, 1);

INSERT INTO Suitability\_List\_Map VALUES (3, 1, NULL);

INSERT INTO Suitability\_List\_Map VALUES (4, 1, 1);

INSERT INTO Suitability\_List\_Map VALUES (5, 1, NULL);

INSERT INTO Suitability\_List\_Map VALUES (1, 2, 2);

INSERT INTO Suitability\_List\_Map VALUES (2, NULL, 2);

INSERT INTO Suitability\_List\_Map VALUES (3, 3, 3);

INSERT INTO Suitability\_List\_Map VALUES (4, 3, NULL);

INSERT INTO Suitability\_List\_Map VALUES (5, 4, 4);

INSERT INTO Suitability\_List\_Map VALUES (2, 4, NULL);

INSERT INTO Suitability\_List\_Map VALUES (3, 5, 5);

INSERT INTO Suitability\_List\_Map VALUES (4, NULL, 6);

INSERT INTO Suitability\_List\_Map VALUES (5, 6, NULL);

INSERT INTO Suitability\_List\_Map VALUES (1, 7, 7);

INSERT INTO Suitability\_List\_Map VALUES (2, NULL, 7);

INSERT INTO Suitability\_List\_Map VALUES (3, 7, 8);

INSERT INTO Suitability\_List\_Map VALUES (4, 8, 9);

INSERT INTO Suitability\_List\_Map VALUES (5, 9, NULL);

INSERT INTO Suitability\_List\_Map VALUES (6, 10, 10);

INSERT INTO Instrument\_List\_Map VALUES (1, 1);

INSERT INTO Instrument\_List\_Map VALUES (4, 1);

INSERT INTO Instrument\_List\_Map VALUES (5, 2);

INSERT INTO Instrument\_List\_Map VALUES (3, 2);

INSERT INTO Instrument\_List\_Map VALUES (2, 2);

INSERT INTO Instrument\_List\_Map VALUES (6, 3);

INSERT INTO Instrument\_List\_Map VALUES (8, 3);

INSERT INTO Instrument\_List\_Map VALUES (7, 4);

INSERT INTO Instrument\_List\_Map VALUES (1, 5);

INSERT INTO Instrument\_List\_Map VALUES (9, 5);

INSERT INTO Instrument\_List\_Map VALUES (10, 5);

INSERT INTO Instrument\_List\_Map VALUES (13, 6);

INSERT INTO Instrument\_List\_Map VALUES (12, 7);

INSERT INTO Instrument\_List\_Map VALUES (11, 7);

INSERT INTO Instrument\_List\_Map VALUES (2, 8);

INSERT INTO Instrument\_List\_Map VALUES (13, 8);

INSERT INTO Instrument\_List\_Map VALUES (7, 8);

INSERT INTO Instrument\_List\_Map VALUES (1, 9);

INSERT INTO Instrument\_List\_Map VALUES (2, 9);

INSERT INTO Instrument\_List\_Map VALUES (3, 10);

INSERT INTO Instrument\_List\_Map VALUES (4, 10);

INSERT INTO Performer\_List\_Map VALUES (1, 1);

INSERT INTO Performer\_List\_Map VALUES (4, 1);

INSERT INTO Performer\_List\_Map VALUES (5, 2);

INSERT INTO Performer\_List\_Map VALUES (3, 2);

INSERT INTO Performer\_List\_Map VALUES (2, 2);

INSERT INTO Performer\_List\_Map VALUES (6, 3);

INSERT INTO Performer\_List\_Map VALUES (8, 3);

INSERT INTO Performer\_List\_Map VALUES (7, 4);

INSERT INTO Performer\_List\_Map VALUES (1, 5);

INSERT INTO Performer\_List\_Map VALUES (9, 5);

INSERT INTO Performer\_List\_Map VALUES (10, 5);

INSERT INTO Performer\_List\_Map VALUES (12, 6);

INSERT INTO Performer\_List\_Map VALUES (12, 7);

INSERT INTO Performer\_List\_Map VALUES (11, 7);

INSERT INTO Performer\_List\_Map VALUES (2, 8);

INSERT INTO Performer\_List\_Map VALUES (12, 8);

INSERT INTO Performer\_List\_Map VALUES (7, 8);

INSERT INTO Performer\_List\_Map VALUES (1, 9);

INSERT INTO Performer\_List\_Map VALUES (2, 9);

INSERT INTO Performer\_List\_Map VALUES (3, 10);

INSERT INTO Performer\_List\_Map VALUES (4, 10);

INSERT INTO Composer\_List\_Map VALUES (1, 1);

INSERT INTO Composer\_List\_Map VALUES (2, 2);

INSERT INTO Composer\_List\_Map VALUES (3, 3);

INSERT INTO Composer\_List\_Map VALUES (4, 4);

INSERT INTO Composer\_List\_Map VALUES (1, 5);

INSERT INTO Composer\_List\_Map VALUES (2, 6);

INSERT INTO Composer\_List\_Map VALUES (3, 7);

INSERT INTO Composer\_List\_Map VALUES (4, 8);

INSERT INTO Composer\_List\_Map VALUES (1, 9);

INSERT INTO Composer\_List\_Map VALUES (2, 10);

INSERT INTO Album\_List\_Map VALUES (1, 3);

INSERT INTO Album\_List\_Map VALUES (2, 6);

INSERT INTO Album\_List\_Map VALUES (3, 9);

INSERT INTO Organisation\_List\_Map VALUES (1, 1);

INSERT INTO Organisation\_List\_Map VALUES (1, 2);

INSERT INTO Organisation\_List\_Map VALUES (3, 3);

INSERT INTO Organisation\_List\_Map VALUES (2, 4);

INSERT INTO Organisation\_List\_Map VALUES (2, 5);

INSERT INTO Organisation\_List\_Map VALUES (4, 6);

INSERT INTO Organisation\_List\_Map VALUES (4, 7);

INSERT INTO Organisation\_List\_Map VALUES (5, 8);

INSERT INTO Organisation\_List\_Map VALUES (5, 9);

INSERT INTO Organisation\_List\_Map VALUES (5, 10);

INSERT INTO Person\_Details VALUES ('Sam', 'Yellow', '01111111111');

INSERT INTO Person\_Details VALUES ('John', 'Green', '01111111112');

INSERT INTO Person\_Details VALUES ('Tony', 'Red', '01111111113');

INSERT INTO Person\_Details VALUES ('Peter', 'White', '01111111114');

INSERT INTO Person\_Details VALUES ('Jackson', 'Blue', '01111111115');

INSERT INTO Person\_Details VALUES ('Charles', 'Mathews', '01111111116');

INSERT INTO Show\_Instance VALUES (1 , 1 ,'20131209','18:00:00');

INSERT INTO Show\_Instance VALUES (2 , 2 ,'20120821','12:00:00');

INSERT INTO Show\_Instance VALUES (3 , 3 ,'20120301','10:00:00');

INSERT INTO Show\_Instance VALUES (4 , 4 ,'20131214','17:00:00');

INSERT INTO Show\_Instance VALUES (5 , 5 ,'20130427','23:30:00');

INSERT INTO Show\_Instance VALUES (6 , 6 ,'20131209','07:00:00');

INSERT INTO Show\_Instance VALUES (6 , 6 ,'20131210','07:00:00');

INSERT INTO Show\_Instance VALUES (6 , 6 ,'20131211','07:00:00');

INSERT INTO Show\_Instance VALUES (7 , 7 ,'20111215','16:00:00');

INSERT INTO Show\_Instance VALUES (8 , 8 ,'20111221','13:30:00');

INSERT INTO Show\_Instance VALUES (9 , 9 ,'20110502','04:45:00');

INSERT INTO Show\_Instance VALUES (10, 10 ,'20110813','22:55:00');

INSERT INTO Show\_Comment\_Map VALUES (1, 1);

INSERT INTO Show\_Comment\_Map VALUES (1, 1);

INSERT INTO Show\_Comment\_Map VALUES (1, 1);

INSERT INTO Show\_Comment\_Map VALUES (2, 2);

INSERT INTO Show\_Comment\_Map VALUES (3, 2);

INSERT INTO Show\_Comment\_Map VALUES (5, 5);

INSERT INTO Show\_Comment\_Map VALUES (6, 4);

INSERT INTO Show\_Comment\_Map VALUES (7, 6);

INSERT INTO Show\_Comment\_Map VALUES (8, 7);

INSERT INTO Show\_Comment\_Map VALUES (9, 7);

INSERT INTO Show\_Comment\_Map VALUES (9, 8);

INSERT INTO Show\_Comment\_Map VALUES (1, 9);

INSERT INTO Show\_Comment\_Map VALUES (8, 10);

INSERT INTO Show\_Comment\_Map VALUES (6, 11);

INSERT INTO Show\_Staff\_Map VALUES (1, 1, 1);

INSERT INTO Show\_Staff\_Map VALUES (3, 1, 0);

INSERT INTO Show\_Staff\_Map VALUES (4, 2, 1);

INSERT INTO Show\_Staff\_Map VALUES (1, 3, 1);

INSERT INTO Show\_Staff\_Map VALUES (4, 4, 0);

INSERT INTO Show\_Staff\_Map VALUES (5, 4, 1);

INSERT INTO Show\_Staff\_Map VALUES (1, 4, 0);

INSERT INTO Show\_Staff\_Map VALUES (2, 5, 1);

INSERT INTO Show\_Staff\_Map VALUES (3, 5, 0);

INSERT INTO Show\_Staff\_Map VALUES (1, 7, 1);

INSERT INTO Show\_Staff\_Map VALUES (2, 6, 1);

INSERT INTO Show\_Staff\_Map VALUES (5, 8, 1);

INSERT INTO Show\_Staff\_Map VALUES (4, 9, 0);

INSERT INTO Show\_Staff\_Map VALUES (3, 9, 1);

INSERT INTO Show\_Staff\_Map VALUES (2, 10, 1);

INSERT INTO Request\_Details VALUES ('A Song', 'Hope This Cheers You All Up', NULL);

INSERT INTO Request\_Details VALUES ('C Song', 'In Memory Of', 'My Brother Andrew');

INSERT INTO Request\_Details VALUES ('D Song', 'How You Have A Nice Day', 'My Son Harry');

INSERT INTO Request\_Details VALUES ('F Song', 'Happy Valentines Day', 'My Wife Emily');

INSERT INTO Request\_Details VALUES ('H Song', 'Love This Song', NULL);

INSERT INTO Request\_Details VALUES ('J Song', 'Who Doesnt Like This Song', NULL);

INSERT INTO Request\_Details VALUES ('B Song', 'ITS CHRISTMAS!!', NULL);

INSERT INTO Request\_Details VALUES ('C Song', 'I Adore This Song', 'Younger Brother Chris');

INSERT INTO Request\_Details VALUES ('A Song', 'I Love Playing This Song', NULL);

INSERT INTO Request\_Details VALUES ('A Song', 'I Adore This Song', 'Younger Brother Chris');

INSERT INTO Request\_Details VALUES ('A Song', 'I Love Playing This Song', NULL)

INSERT INTO Request\_Details VALUES ('Some Thing Only We Know', 'I Love Playing This Song All The Time', NULL)

INSERT INTO Request\_Details VALUES ('Some Thing Only We Know', 'I Adore This Song ALWAYS', 'Paul');

INSERT INTO Request\_Details VALUES ('Some Thing Only We Know', 'I Love Playing This Song ALWAYS', NULL)

INSERT INTO Request\_Details VALUES ('Some Thing Only We Know', 'I Adore This Song Sometimes', 'Chris');

INSERT INTO Request\_Details VALUES ('Some Thing Only We Know', 'I Love Playing This Song Sometimes', NULL)

INSERT INTO Person\_Request\_Map VALUES (1, 1);

INSERT INTO Person\_Request\_Map VALUES (2, 2);

INSERT INTO Person\_Request\_Map VALUES (3, 3);

INSERT INTO Person\_Request\_Map VALUES (4, 4);

INSERT INTO Person\_Request\_Map VALUES (5, 5);

INSERT INTO Person\_Request\_Map VALUES (6, 6);

INSERT INTO Staff\_Request\_Map VALUES (5, 7);

INSERT INTO Staff\_Request\_Map VALUES (3, 8);

INSERT INTO Staff\_Request\_Map VALUES (1, 9);

INSERT INTO Show\_Request\_Map VALUES (1, 1);

INSERT INTO Show\_Request\_Map VALUES (2, 2);

INSERT INTO Show\_Request\_Map VALUES (3, 3);

INSERT INTO Show\_Request\_Map VALUES (4, 4);

INSERT INTO Show\_Request\_Map VALUES (5, 5);

INSERT INTO Show\_Request\_Map VALUES (6, 6);

INSERT INTO Show\_Request\_Map VALUES (7, 7);

INSERT INTO Show\_Request\_Map VALUES (8, 8);

INSERT INTO Show\_Request\_Map VALUES (9, 9);

CREATE FUNCTION getStaffSongPlayCount\_udf (@nStaffFName varchar(35), @nStaffSName varchar(35))

RETURNS TABLE

AS

RETURN

SELECT Count(Song\_Details.song\_Title) AS 'Songs Played' FROM Song\_Details INNER JOIN

Playlist\_Song\_Map ON Song\_Details.song\_Details\_ID = Playlist\_Song\_Map.song\_Details\_ID INNER JOIN

Playlist\_Details ON Playlist\_Song\_Map.playlist\_Details\_ID = Playlist\_Details.playlist\_Details\_ID

Where Playlist\_Song\_Map.staff\_Details\_ID = (SELECT Staff\_Details.staff\_Details\_ID FROM Staff\_Details

WHERE Staff\_Details.staff\_FName = @nStaffFName AND Staff\_Details.staff\_SName = @nStaffSName)

GO

CREATE FUNCTION getNumberDays\_udf (@sDate date)

RETURNS INT

AS

BEGIN

DECLARE @nDay int

SET @nDay = DATEDIFF(day, @sDate, GETDATE());

RETURN @nDay

END

CREATE FUNCTION getSongPlayCount\_udf (@nPlay\_Count int)

RETURNS TABLE

AS

RETURN

(

SELECT song\_Title, song\_Duration, song\_Play\_Count, song\_Added

FROM Song\_Details

WHERE song\_Play\_Count >= @nPlay\_Count

)

CREATE FUNCTION getShowOccurrence\_udf (@nStaffFName varchar(35), @nStaffSName varchar(35))

RETURNS int

AS

BEGIN

DECLARE @nShows int

SET @nShows = (SELECT Count(Show\_Instance.show\_Instance\_ID) AS 'Show Played' FROM Show\_Instance INNER JOIN

Show\_Staff\_Map ON Show\_Instance.show\_Instance\_ID = Show\_Staff\_Map.show\_Instance\_ID

WHERE Show\_Staff\_Map.staff\_Details\_ID = (SELECT Staff\_Details.staff\_Details\_ID FROM Staff\_Details

WHERE Staff\_Details.staff\_FName = @nStaffFName AND Staff\_Details.staff\_SName = @nStaffSName))

RETURN @nShows

END

CREATE VIEW all\_Comments AS

SELECT Comment\_Details.comment\_Message, Comment\_Details.comment\_Type, Show\_Instance.show\_Details\_ID FROM Comment\_Details INNER JOIN

Show\_Comment\_Map ON Comment\_Details.comment\_Details\_ID = Show\_Comment\_Map.comment\_Details\_ID INNER JOIN

Show\_Instance ON Show\_Comment\_Map.show\_Instance\_ID = Show\_Instance.show\_Instance\_ID

CREATE FUNCTION getNumberRequest\_udf ( @rSong varchar (255) )

RETURNS INT

AS

BEGIN

DECLARE @nSong int

SET @nSong = (SELECT COUNT(Request\_Details.request\_Song) From Request\_Details

WHERE Request\_Details.request\_Song = @rSong)

RETURN @nSong

END

**1.3.0 SQL Queries**

**1.3.1 Your Choice Show Query**

SELECT song\_Title AS 'Song Title', song\_Duration AS 'Song Duration', song\_Play\_Count AS 'Song Play Count',stuff ((SELECT distinct ', ' + cast(g.genre\_Type as varchar(255))

FROM Genre\_List\_Map g1 INNER JOIN

Genre\_Details g ON g1.genre\_Details\_ID = g.genre\_Details\_ID

WHERE g1.song\_Details\_ID = s.song\_Details\_ID

FOR XML PATH('')),1,1,'') AS Genres,

stuff ((SELECT distinct ', ' + cast(s2.suitability\_Type as varchar(255))

FROM Suitability\_List\_Map s1 INNER JOIN

Suitability\_Details s2 ON s1.suitability\_Details\_ID = s2.suitability\_Details\_ID

WHERE s1.song\_Details\_ID = s.song\_Details\_ID

FOR XML PATH('')),1,1,'') AS Suitabilitys,

stuff ((SELECT distinct ', ' + cast(i.instrument\_Name as varchar(255))

FROM Instrument\_List\_Map i1 INNER JOIN

Instrument\_Details i ON i1.instrument\_Details\_ID = i.instrument\_Details\_ID

WHERE i1.song\_Details\_ID = s.song\_Details\_ID

FOR XML PATH('')),1,1,'') AS Instruments,

stuff ((SELECT distinct ', ' + cast(c.composer\_FName as varchar(35)) + ' ' + cast(c.composer\_SName as varchar(35))

FROM Composer\_List\_Map c1 INNER JOIN

Composer\_Details c ON c1.composer\_Details\_ID = c.composer\_Details\_ID

WHERE c1.song\_Details\_ID = s.song\_Details\_ID

FOR XML PATH('')),1,1,'') AS Composers,

stuff ((SELECT distinct ', ' + cast(p.performer\_FName as varchar(35)) + ' ' + cast(p.performer\_SName as varchar(35)) + ': ' + cast(p.performer\_Role as varchar(35))

FROM Performer\_List\_Map p1 INNER JOIN

Performer\_Details p ON p1.performer\_Details\_ID = p.performer\_Details\_ID

WHERE p1.song\_Details\_ID = s.song\_Details\_ID

FOR XML PATH('')),1,1,'') AS Performers,

stuff ((SELECT distinct ', ' + cast(a.album\_Name as varchar(35))

FROM Album\_List\_Map a1 INNER JOIN

Album\_Details a ON a1.album\_Details\_ID = a.album\_Details\_ID

WHERE a1.song\_Details\_ID = s.song\_Details\_ID

FOR XML PATH('')),1,1,'') AS Album,

stuff ((SELECT distinct ', ' + cast(o.organisation\_Name as varchar(255)) + ': ' + cast(o.organisation\_Type as varchar(255)) + ', ' + cast(CASE o.organisation\_Local

WHEN 1 THEN 'Local'

ELSE 'Not Local' END as varchar(255)) + ': ' + cast(o.organisation\_Founded as varchar(35))

FROM Organisation\_List\_Map o1 INNER JOIN

Organisation\_Details o ON o1.organisation\_Details\_ID = o.organisation\_Details\_ID

WHERE o1.song\_Details\_ID = s.song\_Details\_ID

FOR XML PATH('')),1,1,'') AS Organisation, song\_Added AS 'Song Added'

FROM Song\_Details s INNER JOIN

Playlist\_Song\_Map ON s.song\_Details\_ID = Playlist\_Song\_Map.song\_Details\_ID INNER JOIN

Playlist\_Details ON Playlist\_Song\_Map.playlist\_Details\_ID = Playlist\_Details.playlist\_Details\_ID INNER JOIN

Show\_Instance ON Playlist\_Details.playlist\_Details\_ID = Show\_Instance.playlist\_Details\_ID

WHERE Show\_Instance.show\_Details\_ID = (SELECT Show\_Details.show\_Details\_ID FROM Show\_Details WHERE Show\_Details.show\_Name = 'Your Choice Show')

**1.3.2 Beaconside Community Choir Query**

SELECT stuff ((SELECT distinct ', ' + cast(o.organisation\_Name as varchar(255))

FROM Organisation\_List\_Map o1 INNER JOIN

Organisation\_Details o ON o1.organisation\_Details\_ID = o.organisation\_Details\_ID

WHERE o1.song\_Details\_ID = s.song\_Details\_ID

FOR XML PATH('')),1,1,'') AS Organisation,

song\_Title AS 'Song Title', song\_Duration AS 'Song Duration',

stuff ((SELECT distinct ', ' + cast(s2.suitability\_Type as varchar(255))

FROM Suitability\_List\_Map s1 INNER JOIN

Suitability\_Details s2 ON s1.suitability\_Details\_ID = s2.suitability\_Details\_ID

WHERE s1.song\_Details\_ID = s.song\_Details\_ID

FOR XML PATH('')),1,1,'') AS Suitabilitys,

stuff ((SELECT distinct ', ' + cast(p.performer\_FName as varchar(35)) + ' ' + cast(p.performer\_SName as varchar(35)) + ': ' + cast(p.performer\_Role as varchar(35))

FROM Performer\_List\_Map p1 INNER JOIN

Performer\_Details p ON p1.performer\_Details\_ID = p.performer\_Details\_ID

WHERE p1.song\_Details\_ID = s.song\_Details\_ID

FOR XML PATH('')),1,1,'') AS Performers,

CONVERT(varchar, DATEADD(ms, SUM(datediff(SECOND, 0, s.song\_Duration)) OVER() \* 1000, 0), 114) AS 'Total Combined Length Of Songs'

FROM Organisation\_Details INNER JOIN

Organisation\_List\_Map ON Organisation\_Details.organisation\_Details\_ID = Organisation\_List\_Map.organisation\_Details\_ID INNER JOIN

Song\_Details s ON Organisation\_List\_Map.song\_Details\_ID = s.song\_Details\_ID

WHERE Organisation\_Details.organisation\_Name = 'Beaconside Community Choir'

**1.3.3 Staff Shows, Songs and Employed Status Query**

SELECT s.staff\_Title, s.staff\_FName, s.staff\_SName, s.staff\_PNumber, s.staff\_Email, s.staff\_Role,

Address\_Details.address\_Postcode, Address\_Details.address\_County ,

Contract\_Details.contract\_Type, Contract\_Details.contract\_SDate, Address\_Details.address\_Street, Address\_Details.address\_City,

dbo.getNumberDays\_udf(Contract\_Details.contract\_SDate) AS 'Days Employed',

CASE

WHEN dbo.getNumberDays\_udf(Contract\_Details.contract\_SDate) < = 120 THEN 'Employed Recently'

WHEN dbo.getNumberDays\_udf(Contract\_Details.contract\_SDate) < = 365 THEN 'Employed Within The Last Years'

WHEN dbo.getNumberDays\_udf(Contract\_Details.contract\_SDate) < = 730 THEN 'Employed Within The Last 2 Years'

WHEN dbo.getNumberDays\_udf(Contract\_Details.contract\_SDate) < = 1095 THEN 'Employed Within The Last 3 Years'

WHEN dbo.getNumberDays\_udf(Contract\_Details.contract\_SDate) < = 1460 THEN 'Employed Within The Last 4 Years'

END AS 'Staff Status' , songsPlayed.\*, dbo.getShowOccurrence\_udf (s.staff\_FName, s.staff\_SName) AS 'Shows Played'

FROM Staff\_Details s

CROSS APPLY

getStaffSongPlayCount\_udf (s.staff\_FName, s.staff\_SName) songsPlayed

INNER JOIN

Staff\_Contract\_Map ON s.staff\_Details\_ID = Staff\_Contract\_Map.staff\_Details\_ID INNER JOIN

Contract\_Details ON Staff\_Contract\_Map.contract\_Details\_ID = Contract\_Details.contract\_Details\_ID INNER JOIN

Address\_Details ON s.address\_Details\_ID = Address\_Details.address\_Details\_ID

WHERE Contract\_Details.contract\_FDate IS NULL AND Contract\_Details.contract\_Type = 'Full Time Permanent'

AND dbo.getShowOccurrence\_udf (s.staff\_FName, s.staff\_SName) >= (SELECT Count(Show\_Instance.show\_Instance\_ID) AS 'Show Played' FROM Show\_Instance) / 4

ORDER BY s.staff\_Details\_ID)

**1.3.4 Comment Length, Status for Weekly Shows Query**

SELECT comment\_Message, comment\_Type ,

LEN(comment\_Message) AS 'Comment Length',

CASE(comment\_Type) WHEN 'Negative' THEN 'Requires Attention' WHEN 'Positive' THEN 'No Attention Required'END AS 'Status'

FROM all\_Comments

WHERE show\_Details\_ID IN (SELECT Show\_Details.show\_Details\_ID FROM Show\_Details WHERE Show\_Details.show\_Type ='Weekly')

ORDER BY all\_Comments.comment\_Message ASC

**1.4.0** **SQL Object Elements**

**1.4.1 Number of Times Played Trigger**

CREATE TRIGGER update\_Play\_Count\_tg

ON Show\_Instance

AFTER INSERT , UPDATE

AS

DECLARE @date varchar(100)

DECLARE @time varchar(100)

DECLARE @cDate varchar(100)

DECLARE @cTime varchar(100)

DECLARE @ID int

SELECT @cDate =(convert(varchar, getdate(), 112))

SELECT @cTime =(convert(varchar, getdate(), 114))

SELECT @date =(SELECT show\_Air\_Date FROM INSERTED)

SELECT @time =(SELECT show\_Air\_Time FROM INSERTED)

SELECT @ID =(SELECT show\_Details\_ID FROM INSERTED)

IF((@date <= @cDate) AND (@time <= @cTime))

BEGIN

UPDATE Song\_Details

SET song\_Play\_Count = song\_Play\_Count + 1

FROM

Show\_Instance LEFT JOIN

Playlist\_Details ON Show\_Instance.playlist\_Details\_ID = Playlist\_Details.playlist\_Details\_ID INNER JOIN

Playlist\_Song\_Map ON Playlist\_Details.playlist\_Details\_ID = Playlist\_Song\_Map.playlist\_Details\_ID INNER JOIN

Song\_Details ON Playlist\_Song\_Map.song\_Details\_ID = Song\_Details.song\_Details\_ID

WHERE Show\_Instance.show\_Instance\_ID = @ID

PRINT 'The Play Count Has Been Updated.';

END

**1.4.2 Populating Playlist with Requests / Dedications Trigger**

CREATE TRIGGER add\_Request\_Playlist\_tg

ON Request\_Details

AFTER INSERT

AS

DECLARE @songName varchar(255);

DECLARE @requestID int;

DECLARE @songID int;

DECLARE @showID int;

DECLARE @playlistID int;

DECLARE @staffID int;

SELECT @requestID =(SELECT request\_Details\_ID FROM INSERTED)

SELECT @songName =(SELECT request\_Song FROM INSERTED)

SET @showID = CASE

WHEN (SELECT request\_Dedication FROM INSERTED) IS NULL THEN 9

ELSE 10

END

IF(@showID IS NOT NULL)

BEGIN

IF EXISTS (SELECT Song\_Details.song\_Title FROM Song\_Details WHERE Song\_Details.song\_Title = @songName)

BEGIN

SET @staffID = (SELECT Show\_Staff\_Map.staff\_Details\_ID FROM Show\_Instance INNER JOIN

Show\_Staff\_Map ON Show\_Instance.show\_Instance\_ID = Show\_Staff\_Map.show\_Instance\_ID

WHERE Show\_Staff\_Map.show\_Instance\_ID =@showID AND Show\_Staff\_Map.show\_Key\_Contact = 1)

SET @playlistID = (SELECT Show\_Instance.playlist\_Details\_ID FROM Show\_Instance INNER JOIN

Playlist\_Details ON Show\_Instance.playlist\_Details\_ID = Playlist\_Details.playlist\_Details\_ID

WHERE Show\_Instance.show\_Details\_ID = @showID)

SET @songID =

(SELECT Song\_Details.song\_Details\_ID FROM Song\_Details

WHERE Song\_Details.song\_Title = @songName)

INSERT INTO Show\_Request\_Map VALUES (@requestID, @showID);

INSERT INTO Playlist\_Song\_Map VALUES (@playlistID, @songID, @staffID);

PRINT 'The Request Has Been Added To A Playlist.';

END

END

**1.4.2 Most Request Songs Not In the Database Procedure**

CREATE PROCEDURE create\_Most\_Request\_proc @sName varchar(255), @nRequest int

AS

DECLARE @nSong int

IF EXISTS (SELECT Song\_Details.song\_Title FROM Song\_Details WHERE Song\_Details.song\_Title = @sName)

BEGIN

PRINT 'Song Already In Database'

END

ELSE IF EXISTS(SELECT Request\_Details.request\_Song FROM Request\_Details WHERE Request\_Details.request\_Song = @sName)

BEGIN

EXEC @nSong = getNumberRequest\_udf @rSong= @sName

IF (@nSong > = @nRequest)

BEGIN

IF object\_id('Most\_Request\_NoDB', 'U') is not null

BEGIN

INSERT INTO Most\_Request\_NoDB VALUES (@sName, DEFAULT);

PRINT 'Adding Song Request To Most\_Request\_NoDB'

END

ELSE

BEGIN

CREATE TABLE Most\_Request\_NoDB

(

most\_Request\_NoDB\_ID int IDENTITY(1,1) NOT NULL,

request\_Song varchar (255) NOT NULL,

request\_Date date DEFAULT getDate(),

CONSTRAINT pk\_MostRequestID PRIMARY KEY (most\_Request\_NoDB\_ID),

)

PRINT 'Most\_Request\_NoDB Table Created'

END

END

END

ELSE

BEGIN

PRINT 'This Song Has Never Been Requested'

END

GO

EXEC create\_Most\_Request\_proc 'Some Thing Only We Know', 4

SELECT \* FROM Most\_Request\_NoDB

**1.4.2 Backup Popular Songs Played More than X Times Procedure**

CREATE PROCEDURE backup\_Popular\_Songs\_proc @nPlay\_Count int

AS

BEGIN

DECLARE @BACKUP\_Songs\_Title varchar (255)

DECLARE @BACKUP\_Songs\_Duration time

DECLARE @BACKUP\_Songs\_Play\_Count int

DECLARE @BACKUP\_Songs\_System date

IF object\_id('Popular\_Songs\_Backup', 'U') is null

BEGIN

CREATE TABLE Popular\_Songs\_Backup

(

popular\_Songs\_Backup\_ID int IDENTITY(1,1) NOT NULL,

popular\_Songs\_Title varchar (255) NOT NULL,

popular\_Songs\_Duration time NOT NULL,

popular\_Songs\_Play\_Count int DEFAULT 0,

popular\_Songs\_System date,

popular\_Songs\_Backup date DEFAULT getDate(),

CONSTRAINT pk\_PopularSongID PRIMARY KEY (popular\_Songs\_Backup\_ID),

)

PRINT 'Popular\_Songs\_Backup Table Created'

END

DECLARE backup\_Popular\_Songs\_csr CURSOR

FOR

SELECT \* FROM getSongPlayCount\_udf(@nPlay\_Count)

OPEN backup\_Popular\_Songs\_csr

FETCH NEXT FROM backup\_Popular\_Songs\_csr

INTO @BACKUP\_Songs\_Title,

@BACKUP\_Songs\_Duration,

@BACKUP\_Songs\_Play\_Count,

@BACKUP\_Songs\_System

WHILE @@FETCH\_STATUS = 0

BEGIN

INSERT INTO Popular\_Songs\_Backup values(

@BACKUP\_Songs\_Title,

@BACKUP\_Songs\_Duration,

@BACKUP\_Songs\_Play\_Count,

@BACKUP\_Songs\_System,

DEFAULT)

FETCH NEXT FROM backup\_Popular\_Songs\_csr

INTO @BACKUP\_Songs\_Title,

@BACKUP\_Songs\_Duration,

@BACKUP\_Songs\_Play\_Count,

@BACKUP\_Songs\_System

END

CLOSE backup\_Popular\_Songs\_csr

DEALLOCATE backup\_Popular\_Songs\_csr

END

EXEC backup\_Popular\_Songs\_proc 4

SELCT \* FROM Popular\_Songs\_Backup

**2.1.0 MongoDB Design**

**2.1.0 Pseudo Code for the Customer Comments Log**

Create CommentsLog

{

Personal:

{

firstName: TEXT,

secondName:TEXT,

DOB: TEXT,

addressOne: TEXT,

addressTwo: TEXT,

city: TEXT,

},

Contact:

{

postcode: TEXT,,

phone: [Phone 1, Phone 2]

email: TEXT,

},

comments: [

{

subject: TEXT,

message: TEXT,

dateCreated: DATE\_TIME,

isReported: TEXT

},

{

subject: TEXT,

message: TEXT,

dateCreated: DATE\_TIME,

isReported: TEXT

}

]

}

**2.2.0 MongoDB Implementation**

**2.2.0 Proof of Concept by Implementing using MongoDB to meet requirements**

cd c:\

C:\MongoDB\bin\mongo.exe

use CustomerCommentLog

db.post.insert({Personal:{firstName: 'Simon',secondName: 'Peterson',DOB: '24:23:1393',addressOne: '24',addressTwo: 'Zoo Lane',city: 'Cupcake

Town'},comments: [{subject: 'MY F\*\*KING 2 year can play better Music',message: 'This Radio Station is a joke please stop spamming adverts and playing

really bad music i will take action!!',dateCreated: new Date('Mar 14, 2013'),isReported: 'Yes'}]});

db.post.insert({comments: [{message: 'I WILL MAKE YOU SUFFER',isReported: 'Yes'}]});

db.post.insert({Personal:{firstName: 'DIE',secondName: 'NOW'},comments: [{subject: 'BOMB BOMB BOMB',message: 'Thnk BOMB',isReported: 'Yes'},

{subject: 'Burning Radio Station',message: 'Oooops I Dropped A Match',isReported: 'Yes'}]});

db.post.insert({Personal:{firstName: 'Harry',secondName: 'Clewlow',DOB: '14:03:1993',addressOne: '122',addressTwo: 'Corp Street',city:

'Stafford',postcode: 'ST16 3LS'},Contact: {phone: ['01777777777', '03777777778'] ,email: 'Harry.Clewlow.1993@gmail.com'},comments: [{subject: 'I Love

This Show',message: 'You Make My Day',dateCreated: new Date('Jun 23, 2012'),isReported: 'No'},{subject: 'Nice Show ',message: 'Good Show Guys Its

Alot Better',dateCreated: new Date('Jun 23, 2012'),isReported: 'No'}]});

//finding all in collection

db.post.find().pretty();

//finding the comments by one person

db.post.find({"Personal.firstName" : "Harry" ,"Personal.secondName" : "Clewlow"}).pretty();

//finding the comments that have been reported

db.post.find({ comments: { $elemMatch: { isReported: "Yes"}}}).pretty();

**2.3.0 Reflection Between Relational and NoSQL**

**2.3.0 Reflection of Relational and NoSQL Design as Shown by MongoDB**

For this assignment I was tasked with designing both a relational and NoSQL database, in doing so I have observed several advantages and disadvantages for both.

When designing and implementing a NoSQL or relational database such as MongoDB and SQL Server you are faced with factors such as elastic scaling which makes it best suited for big data. In comparison a relational data scales every differently where in order to scale you would have to invest in large, more efficient servers, which makes a relational database isn’t a viable choice. Most NoSQL databases support auto-sharding, which allows the data to be speed across servers without an application being aware of the server pool, this means that the query load can be balanced across the servers automatically, and if a server fails if can be replaced with interrupting the application, this gives NoSQL a huge advantage over relational databases which require the application to be information of the server pools and a server fail would cause the application to be down while the error is fixed.

NoSQL in comparison to a relational database is very flexible because it doesn’t have a specific data model which like in the comments system you can store data in any structure you want such as a comment where someone hasn’t left there contact details, personal details or have refused to put their data of birth.

Relational database are considerably hard and more complex to design and implement as shown in the assignment the design for a relational database requires relationship between tables, more management when it comes to data distribution and data models of a NoSQL database. Another difference between NoSQL and relational databases are the types of servers each will use, where NoSQL can run on a cheap device or server in comparison to relational database which require expensive servers and storage, also that fact that NoSQL is open source.

A major disadvantage to NoSQL is that is mainly used for storage and offers little in the way of functionality, as shown in the assignment where you can store many comments, but due to MongoDB not having a function that allows you to return an individual array element such as one comment from a customer that has made more than one negative or positive comment, the function will only return the array that contains a reported comment not the individual array element. When you compare this to a relational database you can perform a variety of functions on the data that is stored.

Due to unstructured nature of NOSQL the consistency of the data might become a concern depending what you are using it for. In the assignment this is not a concern because we are dealing with a mass of unstructured data, where customers can leave as many or as little details about themselves or a comment, but in a situations like the relational database for the radio stations playlist it becomes a massive weakness because you need to keep the same information on all the songs and playlist etc. e.g. you can’t have a one playlist where it stored to songs, who played it and the date when you have one with only the songs on the playlist not who played which song or when it was played.

In conclusion both NoSQL and relational databases are acceptable depending what the database is going to be used for due to NoSQL having unstructured data it works perfectly for storing big data, but if you are wanting to preforming analytics on the data you have stored then you are better off using a relational database to preform advanced queries and data manipulation.