**Database Development**

**(COMP08002)**

**Coursework**

**Solo Version**

Group work: Individual Work (1 student)

Contribution: 60% of final mark

Submission date: Upload by 11:59 pm Friday 17th November (Week 10)

**Database Development (COMP08002)**

**Coursework for *Stream2U* Case Study (Extended)**

**Solo Version**

|  |  |
| --- | --- |
| **Student Name** | **BannerID** |
| Corey Black | B01651145S |

|  |  |
| --- | --- |
| **Database Name** | **Location** |
| B01651145-Stream2U | Student Laptop |

|  |  |  |
| --- | --- | --- |
| **Lecturer Name** | **My Campus** | |
| Tahir Mahmood | Paisley |

1. **ER model for *Stream2U***

***Diagram

Description automatically generated***

**2. Relational Schema for *Stream2U* Database**

**tblMembershipType** (mTypeID, mTypeName, minFilmsStream, mBenefit)

**Primary Key** mTypeID

DELETE NO ACTION, UPDATE NO ACTION

**tblMember** (memberID, firstName, lastName, dob, gender, dateJoined, age, fullName, mTypeID)

**Primary Key** memberID

**Foreign Key** mTypeID references tblMembershipType(mTypeID) ON DELETE CASCADE ON UPDATE CASCADE

**Calculated** age Calculated: fullName tblAccount

**tblWatchList** (watchItemID, dateAdded, dateRemoved)

**Primary Key** watchItemID

**tblFilm** (filmID, title, genre, classification, rentalCharge, duration, synopsis, releaseDate)

**Primary Key** filmID

**tblStream** (streamID, memberID, filmID, memberRating, requestView, startView, requestToStart)

**Primary Key** streamID

**Calculated** requestToStart

**Foreign Key** memberID references tblMember(memberID)

**Foreign Key** filmID references tblFilm(filmID)

**tblFilmTrailer** (filmTrailerID, trailerType, duration, releaseDate)

**Primary** **Key** filmTrailerID

**tblReview** (reviewID, filmRating, filmReview, reviewDate)

**Primary Key** reviewID

**tblAccount** (accountID, createDate, eMail, passwordHash, passwordSalt)

**Primary Key** accountID

**tblComment** (commentID, commentText, commentDate)

**Primary** **Key** commentID

**tblActor** (actorID, name, gender, nationality, twitterName)

**Primary Key** actorID

**3. Data Dictionary for *Stream2U* Database**

The **MembershipType** Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Meaning** | **Data Type** | **Nulls** | **PK / FK** | **Default**  **Value** | **Constraint /Column Property (including data classification)** |
| mTypeID | Unique identifier of type | tinyint | No | PK |  | (Private) |
| mTypeName | Name of membership type | nvarchar(8) | No |  |  | (Public)  Only values ‘Bronze’, ‘Silver’, ‘Gold’, ‘Platinum’ |
| minFilmsStream | Minimal number of films streamed to qualify | smallint | No |  |  | (Public) |
| mBenefit | Describes the benefit achieved on streaming more than a minimum number of films | nvarchar(100) | No |  |  | (Public) |

The **tblMember** table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Meaning** | **Data Type** | **Nulls** | **PK / FK** | **Default**  **Value** | **Constraint /Column Property** |
| memberID | Uniquely identifies each member | int | No | PK |  | (Private)  Set identity specification seed as 1 and increment as 1. |
| firstName | First name of member | nvarchar (30) | No |  |  | (Private) |
| lastname | Last name of member | nvarchar (30) | No |  |  | (Private) |
| dob | Date of birth of member | date | No |  |  | (Private) |
| gender | Gender of member | nchar(1) | Yes |  | ‘F’ | (Private)  Only hold ‘F’, ‘M’ or ‘U’ (U for unassigned) |
| dateJoined | Date member joined | date | No |  | getDate() | (Private) |
| Age | Age of the member | Int | No |  |  | (Private)  Use DATADIFF() function and a trigger |
| fullName | Displays first and last name of member together. |  |  |  |  | (Private)  FullName AS (FirstName + ' ' + LastName) |
| mType | Identifies the member’s current membership type | tinyint | No | FK | 1 | (Private) |

The **tblFilm** table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Meaning** | **Data Type** | **Nulls** | **PK / FK** | **Default**  **Value** | **Constraint /Column Property** |
| filmID | Uniquely identifies each film | int | No | PK |  | (Private)  Set identity specification seed as 101 and increment as 1. |
| title | Title of film | nvarchar  (50) | No |  |  | (Public) |
| classification | British Board classification of film | nvarchar (3) | No |  | 12A | (Public)  Only hold: ‘U’, ‘PG’, ‘12A’, ‘15’ or ‘18’ |
| rentalCharge | Nominal charge to stream a film. Used by company to calculate monetary value of films streamed. | money | No |  | 3.00 | (Private)  Only hold between 1.00 and 10.00 |
| duration | Duration of the movie | int | Allowed |  |  | (Public) |
| synopsis | Brief synopsis of the movie | Text  (Variablelengthtext) | Allowed |  |  | (Public) |
| releaseDate | Release date of the movie | date | Allowed |  |  | (Public) |

The **tblWatchlist** Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Meaning** | **Data Type** | **Nulls** | **PK / FK** | **Default**  **Value** | **Constraint /Column Property** |
| watchItemID | Uniquely identifies for each watchlist item. | int | No | PK |  | (Private)  Set identity specification seed as 1 and increment as 1. |
| dateAdded | The date when a film was added to a watchlist | Datetime | No |  | getDate() | (Private) |
| dateRemoved | The date when a film was removed from a wachlist. | Datetime | No |  | getDate() | (Private) |
| memberID | The ID of the member. | Int | Allowed | FK |  | (Private)  References  tblMember(memberID) |
| filmID | The ID of the film. | Int | Allowed | FK |  | (Private)  references tblFilm(filmID) |

The **tblStream** Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Meaning** | **Data Type** | **Nulls** | **PK / FK** | **Default Value** | **Add Constraint/Column Property** |
| streamID | Uniquely identifies each request to stream a film. | int | No | PK |  | (Private)  Set identity specification seed as 1001 and increment as 1. |
| memberID | Identifies member requesting film. | int | Allowed | FK |  | (Private) |
| filmTrailerID | A unique identifier for each film trailer. | Int | No | FK |  | (Private) |
| filmID | Identifies film to be streamed. | int | No | FK |  | (Private) |
| memberRating | Member’s rating of film from 0.5 star (very poor) to 10 stars (brilliant). | numeric(3,1) | Yes |  | 5 | (Public)  Only values 0.5 to 10.0 (Constraint). |
| requestView | Date and time that member made request. | datetime | No |  | getDate() | (Private) |
| startView | Date and time that member started to view film. | datetime | No |  |  | (Private) |
| requestToStart | Time (in seconds) between member’s request to rent (stream) film and member starting to watch film. | time(6) |  |  |  | (Private)  Use DATEDIFF() function and persistent storage. |

The **tblFilmTrailer** Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Meaning** | **Data Type** | **Nulls** | **PK / FK** | **Default**  **Value** | **Constraint /Column Property** |
| filmTrailerID | A unique identifier for the films trailer. | int | No | PK |  | (Private)  Set identity specification seed as 1 and increment as 1. |
| trailerType | Type of trailer either vide |  |  |  |  |  |
| duration | The length of the trailer. | time | No |  |  | (Public) |
| releaseDate | The date when a film was removed from a wachlist. | Datetime | No |  | getDate() | (Public) |

The **tblActor** Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Meaning** | **Data Type** | **Nulls** | **PK / FK** | **Default Value** | **Add Constraint/Column Property** |
| actorID | A unique identifier for each actor. | int | No | PK |  | (Private) |
| Name | The actor/actress’s name. | Nvarchar(30) | Allowed |  |  | (Private) |
| Gender | The gender of the actor. | Nchar(1) | Allowed |  |  | (Private) |
| Nationality | The nationality of the actor/actress. | int | No |  |  | (Private) |
| twitterName | The actors twitter name if they have one. | Nvarchar(30) | Allowed |  | 5 | (Public) |

The **tblAccount** Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Meaning** | **Data Type** | **Nulls** | **PK / FK** | **Default Value** | **Add Constraint/Column Property** |
| accountID | A unique ID given to every account | int | No | PK |  | (Private)  Set identity specification seed as 1 and increment as 1. |
| createDate | Record the date account was established. | Date | No |  | getDate() | (Private) |
| eMail | Email address linked to the users account | Varchar | No |  |  | (Private) |
| passwordHash | A hashed variation of the accounts password for security. | Varchar or binary | No |  |  | (Private)  passwordHash = Hash(password |
| passwordSalt | A random value used in combination with the password to enhance security. | Varchar or binary | No |  |  | (Private)  Salt = GenerateRandomSalt() |
| memberID | The ID of the account owner. | Int | No | FK |  | (Private)  References tblMember(memberID) |

The **tblReview** Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Meaning** | **Data Type** | **Nulls** | **PK / FK** | **Default Value** | **Add Constraint/Column Property** |
| reviewID | A unique ID given to every review | int | No | PK |  | (Private)  Set identity specification seed as 1 and increment as 1. |
| filmRating | A rating given to the film from 0.5 to 10. | Numeric(3, 1) | No |  | 5 | (Public)  Only values 0.5 to 10.0 (Constraint). |
| filmReview | The user can leave a review about the film. | Text(variablelengthtext) | Allowed |  |  | (Public) |
| reviewDate | The date the review was created. | date | No |  |  | (Public) |

The **tblComment** Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Meaning** | **Data Type** | **Nulls** | **PK / FK** | **Default Value** | **Add Constraint/Column Property** |
| commentID | A unique identifier for the comment. | int | No | PK |  | (Private) |
| commentText | A section for the writer to leave a comment. | Text(variablelenngthtext) | No |  |  | (Public) |
| commentDate | The date the comment was written. | Text(variablelengthtext) | No |  | getDate() | (Public) |

**4. Build the *Stream2U*database**

**The Stream2U Database (shown in Table View – Standard)**

A screenshot of a computer

Description automatically generated

**The Stream2U Database (shown in Table View – Column Names)**

A screenshot of a computer

Description automatically generated

**5. Add Records into the *Stream2U*database**

**SELECT \* FROM** tblMembershipType

**A table with text and numbers

Description automatically generated**

**SELECT \* FROM** tblMember

**A table of data with numbers and letters

Description automatically generated**

**SELECT \* FROM** tblWatchList

**A screenshot of a data

Description automatically generated**

**SELECT \* FROM** tblFilm

**A screenshot of a white background

Description automatically generated**

**SELECT \* FROM** tblStream

**A screenshot of a table

Description automatically generated**

**SELECT \* FROM** tblFilmTrailer

**A screenshot of a screen

Description automatically generated**

**SELECT \* FROM** tblAccount

**A close-up of a text

Description automatically generated**

**SELECT \* FROM** tblReview

**A screenshot of a computer

Description automatically generated**

**SELECT \* FROM** tblComment

**A screenshot of a text box

Description automatically generated**

**SELECT \* FROM** tblActor

**A table of names with black text

Description automatically generated**

**6. Querying your *Stream2U* database**

**Query 1**

**Action**: create the ‘tblStream’ table, specifying column names and data types.

**Purpose**: This query is useful for setting specific data types to columns, and is a fast method of utilizing SQL to write your tables instead of using the SSMS table creator.

USE B01651145\_Stream2U; //Specifying the Database will ensure the table is created within it.

CREATE TABLE tblStreamExample ( //CREATE TABLE command used with the table name following

streamID int, //Integer datatype used

memberID int,

filmID int,

memberRating numeric(3,1), //Numeric datatype used specifying decimal points

requestView datetime NULL, //datetime datatype and specify column as ALLOW NULL

startView datetime,

requestToStart AS (DATEDIFF(SECOND, requestView, startView)) //Use the DATEDIFF command to get the difference between the columns ‘requestView & startView’ and store in ‘requestToStart’ column

);

**A screenshot of a computer

Description automatically generated**

**Query 2**

**Action**: create the ‘tblMember’ table, specifying column names and data types.

**Purpose**: By creating the table with SQL I can specify the ‘fullName’ column to use the values stored in ‘firstName’ & ‘lastName’.

USE B01651145\_Stream2U; //This command ensures the table is created within the correct database

CREATE TABLE tblMemberExample ( //CREATE TABLE is used like before, followed by the table name.

memberID INT, //Integer datatype

firstName NVARCHAR(30), //NVARCHAR(30) datatype

lastName NVARCHAR(30),

dob DATE, //Date datatype

gender NCHAR(1) NULL, //NCHAR(1) datatype allowing NULL

dateJoined DATE,

age NVARCHAR(60),

fullName AS (firstName + ' ' + lastName) //This command combines the firstName and lastName values to give the fullName in this column

);

**A screenshot of a computer program

Description automatically generated**

**Query 3**

**Action**: SELECT memberID, filmID, title, memberRating, requestView, startView and requestToStart from ‘tblStream’ and JOIN ‘filmID’ from tblFilm, where memberID is targetMemberID. Output in descending order.

**Purpose**: By creating the table with SQL I can specify the ‘fullName’ column to use the values stored in ‘firstName’ & ‘lastName’.

DECLARE @targetMemberID INT; //Declare the targetMemberID as an INT to use later

SET @targetMemberID = 1337; //Set targetMemberID as the ID we’re looking for

SELECT //Select these columns specified

s.memberID,

s.filmID,

f.title AS FilmTitle,

s.memberRating,

s.requestView,

s.startView,

s.requestToStart

FROM //From tblStream

tblStream s

JOIN //Join filmID from tblFilm

tblFilm f ON s.filmID = f.filmID

WHERE //Where memberID matches targetMemberID

s.memberID = @targetMemberID

ORDER BY //Output in descending order

s.startView DESC;

**A screenshot of a computer

Description automatically generated**

**Query 4**

**Action**: SELECT watchItemID, dateAdded, dateRemoved, filmID and memberID from tblWatchList, join memberID from tblMember and output the records WHERE memberID = 1340.

**Purpose**: This query is useful for someone accessing the database to call what movies a user has watched and dates etc.

SELECT //SELECT these records specified

w.watchItemID,

w.dateAdded,

w.dateRemoved,

w.filmID,

w.memberID

FROM //FROM tblWatchList

tblWatchList w

JOIN //JOIN memberID from tblMember

tblMember m ON w.memberID = m.memberID

WHERE //WHERE memberID = 1340

m.memberID = 1340;

**A screenshot of a computer

Description automatically generated**

**Query 5**

**Action**: UPDATE a record within a table, in this case UPDATE the ‘twitterName’ of an actor from the ‘tblActor’ table where twitterName = “example name”, and UPDATE to “updated example name”

**Purpose**: This query allows the database analysist to update records upon changes which can be useful in multiple situations.

SELECT //SELECT actorID, name, and twitterName

actorID,

name,

twitterName AS PreviousTwitterName, //Store twitterName as previousTwitterName to display changes in table produced

'Clovis\_cormier\_real' AS UpdatedTwitterName

FROM //FROM tblActor

tblActor

WHERE //WHERE twitterName = “specified name”

twitterName = 'Clovis\_cormier-online';

UPDATE //UPDATE command for tblActor

tblActor

SET //SET twitterName as “updated name”

twitterName = 'Clovis\_cormier\_real'

WHERE //WHERE twitterName = “old name”

twitterName = 'Clovis\_cormier-online';

**A screenshot of a computer program

Description automatically generated**

**Query 6**

**Action**: INSERT a record into the tblStream table and specify the values

**Purpose**: This allows someone to add records to an existing table using SQL, the user can specify the values of each column by declaring them within the VALUES command. This is useful in numerous ways and we can update/delete the data later.

INSERT INTO tblStream (streamID, memberID, filmID, memberRating, requestView, startView) //INSERT INTO ‘tblStream’ the records specified within parenthesis

VALUES ( //Declare the values of each column on a new line

2029,

1345,

40,

5.6,

'2023-11-21 15:30',

'2023-11-21 15:45'

);

**A screenshot of a computer

Description automatically generated**

**Query 7**

**Action**: DELETE records from a table where a memberRating is below a specific rating.

**Purpose**: This is useful for deletion of multiple records at a time that match a specific criteria, in this example the records that have a member rating of below 4 have been removed from the database.

SELECT streamID, filmID, memberRating //SELECT streamID, filmID and memberRating from the table before deleting the records

FROM tblStream //FROM tblStream

WHERE memberRating < 4; //Condition of memberRating being less than 4

DELETE FROM tblStream //DELETE the record by using DELETE FROM ‘table’

WHERE memberRating < 4; //WHERE ‘condition is met’

**A screenshot of a computer

Description automatically generated**

**Query 8**

**Action**: ALTER and existing table to add a new column, and specify the datatype of the column.

**Purpose**: This can be used to add new columns to multiple tables or just an easy method of adding the column to a table if there is a specific datatype that needs to be programmed etc.

ALTER TABLE tblComment //ALTER table ‘specified table’

ADD authorID INT; //ADD ‘new column’ and ‘datatype’

**A screenshot of a computer

Description automatically generated**

**7. Coursework Marking Scheme**

|  |  |
| --- | --- |
| **Section** | **Marks (Out of 100)** |
| Relational Schema | 20 |
| Data Dictionary | 20 |
| Create Database | 30 |
| Insert Records | 10 |
| SQL Statements | 20 |

**8. Submit Your Coursework**

You should save this coursework as a PDF document and upload it using a link (which will appear) in the Formal Assessment 2 – Coursework folder.