Danilo Zelenovic

501032542

No errors in java project VisualNovelDatabase occur, as seen below, with all queries working, with code attached below. The main issues stemmed from jar files not being connected at the beginning. There are no known issues that need to be accounted for as of now. Permission was given in the lab section to re-submit after deadline without penalties.

A screenshot of a computer

Description automatically generated

Dump of VNDB:

BEGIN TRANSACTION;

CREATE TABLE Novel (NovelID INTEGER PRIMARY KEY AUTOINCREMENT, Title VARCHAR(255) NOT NULL, ReadingStatus VARCHAR(50), TimeToFinish REAL, Rating REAL, AgeRating VARCHAR(10) NOT NULL, ReleaseDate DATE NOT NULL, Status VARCHAR(50) NOT NULL, DateAdded DATE);

INSERT INTO Novel VALUES(1,'The House in Fata Morgana','Finished',34.29999999999999715,8.5,'17+','2019-05-28','Completed','2023-03-21');

INSERT INTO Novel VALUES(2,'CLANNAD','Playing',78.0,NULL,'15+','2019-07-04','Completed','2023-06-22');

INSERT INTO Novel VALUES(3,'Class of 09','Finished',3.299999999999999823,7.0,'17+','2022-01-06','Completed','2024-01-12');

INSERT INTO Novel VALUES(4,'Doki Doki Literature Club!','Stalled',6.200000000000000177,NULL,'17+','2021-06-30','Completed','2023-04-12');

INSERT INTO Novel VALUES(5,'Fate/EXTRA','Playing',27.5,NULL,'12+','2011-11-01','Completed','2022-06-18');

INSERT INTO Novel VALUES(6,'Fate/Hollow Ataraxia','Finished',50.0,9.199999999999999289,'17+','2017-10-06','Completed','2022-09-12');

INSERT INTO Novel VALUES(7,'Fate/Stay Night','Finished',87.59999999999999431,9.80000000000000072,'18+','2004-03-26','Completed','2021-07-20');

INSERT INTO Novel VALUES(8,'The Fruit of Grisaia','Playing',75.20000000000000284,9.099999999999999645,'18+','2015-08-28','Completed','2021-10-02');

INSERT INTO Novel VALUES(9,'Wonderful Everyday','Completed',50.5,9.0,'18+','2017-08-30','Completed','2024-01-27');

INSERT INTO Novel VALUES(10,'The Coffin of Andy and LeyLey','Completed',15.5,8.199999999999999289,'13+','2024-01-12','Ongoing','2024-01-29');

INSERT INTO Novel VALUES(11,'Tsukihime -A Piece Of Blue Glass Moon-','Completed',60.0,10.0,'15+','2023-06-17','Ongoing','2023-09-21');

CREATE TABLE Author (AuthorID INTEGER PRIMARY KEY AUTOINCREMENT, AuthorName VARCHAR(255) NOT NULL, Birthdate DATE, CONSTRAINT ck\_author UNIQUE (AuthorName));

INSERT INTO Author VALUES(1,'Maeda Jun','1975-01-03');

INSERT INTO Author VALUES(2,'Max Field',NULL);

INSERT INTO Author VALUES(3,'Dan Salvato','1991-12-23');

INSERT INTO Author VALUES(4,'Hanada Keika',NULL);

INSERT INTO Author VALUES(5,'Nasu Kinoko','1973-11-28');

INSERT INTO Author VALUES(6,'Fujisaki Ryuuta',NULL);

INSERT INTO Author VALUES(7,'Nemlei',NULL);

INSERT INTO Author VALUES(8,'Chitose',NULL);

INSERT INTO Author VALUES(9,'SCA-DI','1986-12-20');

INSERT INTO Author VALUES(10,'Tanaka Kunimitsu',NULL);

CREATE TABLE VoiceActor (VoiceActorID INTEGER PRIMARY KEY AUTOINCREMENT, VoiceName VARCHAR(255) NOT NULL, Birthdate DATE, Gender VARCHAR(10));

INSERT INTO VoiceActor VALUES(1,'Kanemoto Ryousuke','1988-09-09','M');

INSERT INTO VoiceActor VALUES(2,'Hasegawa Ikumi','1980-05-31','F');

INSERT INTO VoiceActor VALUES(3,'Kei Mizusawa','1975-09-14','F');

INSERT INTO VoiceActor VALUES(4,'Nakamura Keisuke','1981-12-16','M');

INSERT INTO VoiceActor VALUES(5,'Ryouko Tanaka','1973-07-28','F');

INSERT INTO VoiceActor VALUES(6,'Taguchi Hiroko','1974-03-29','F');

INSERT INTO VoiceActor VALUES(7,'Kawasumi Ayako','1976-03-30','F');

INSERT INTO VoiceActor VALUES(8,'Sugiyama Noriaki','1976-03-09','M');

INSERT INTO VoiceActor VALUES(9,'Kayli Mills','1994-12-10','F');

INSERT INTO VoiceActor VALUES(10,'Elsie Lovelock','1992-06-10','F');

INSERT INTO VoiceActor VALUES(11,'Saito Chiwa','1981-03-12','F');

CREATE TABLE Composer (ComposerID INTEGER PRIMARY KEY AUTOINCREMENT, ComposerName VARCHAR(255) NOT NULL, Birthdate DATE, Gender VARCHAR(10), CONSTRAINT ck\_composer UNIQUE (ComposerName));

INSERT INTO Composer VALUES(1,'Haga Keita',NULL,'M');

INSERT INTO Composer VALUES(2,'Hideyuki Fukasawa','1970-11-30','M');

INSERT INTO Composer VALUES(3,'Nemlei',NULL,'F');

INSERT INTO Composer VALUES(4,'ryo','1976-10-21','M');

INSERT INTO Composer VALUES(5,'Agematsu Noriyasu','1978-03-01','M');

INSERT INTO Composer VALUES(6,'James Harris',NULL,'M');

INSERT INTO Composer VALUES(7,'temtem',NULL,'F');

INSERT INTO Composer VALUES(8,'Dan Salvato','1991-12-23','M');

INSERT INTO Composer VALUES(9,'Namikawa Soutarou',NULL,'M');

INSERT INTO Composer VALUES(10,'Tsutsumi Yusuke',NULL,'M');

INSERT INTO Composer VALUES(11,'Maeda Jun','1975-01-03','M');

CREATE TABLE Genre (GenreID INTEGER PRIMARY KEY AUTOINCREMENT, GenreName VARCHAR(255) NOT NULL, CONSTRAINT ck\_genre UNIQUE (GenreName));

INSERT INTO Genre VALUES(1,'Kinetic');

INSERT INTO Genre VALUES(2,'Routes');

INSERT INTO Genre VALUES(3,'Horror');

INSERT INTO Genre VALUES(4,'Psychological Horror');

INSERT INTO Genre VALUES(5,'Action');

INSERT INTO Genre VALUES(6,'Romance');

INSERT INTO Genre VALUES(7,'Insanity');

INSERT INTO Genre VALUES(8,'Gore');

INSERT INTO Genre VALUES(9,'ADV');

INSERT INTO Genre VALUES(10,'NVL');

INSERT INTO Genre VALUES(11,'Religious Themes');

INSERT INTO Genre VALUES(12,'Female Protagonist');

INSERT INTO Genre VALUES(13,'Male Protagonist');

INSERT INTO Genre VALUES(14,'Comedy');

CREATE TABLE Platform (PlatformID INTEGER PRIMARY KEY AUTOINCREMENT, PlatformName VARCHAR(255) NOT NULL, CONSTRAINT ck\_platform UNIQUE (PlatformName));

INSERT INTO Platform VALUES(1,'MacOS');

INSERT INTO Platform VALUES(2,'Linux');

INSERT INTO Platform VALUES(3,'Playstation 4');

INSERT INTO Platform VALUES(4,'Playstation 5');

INSERT INTO Platform VALUES(5,'Xbox One');

INSERT INTO Platform VALUES(6,'Xbox Series X|S');

INSERT INTO Platform VALUES(7,'3DS');

INSERT INTO Platform VALUES(8,'Nintendo Switch');

INSERT INTO Platform VALUES(9,'GBA');

INSERT INTO Platform VALUES(10,'PSP');

INSERT INTO Platform VALUES(11,'PSVita');

INSERT INTO Platform VALUES(12,'Windows');

INSERT INTO Platform VALUES(13,'Android');

CREATE TABLE Publisher (PublisherID INTEGER PRIMARY KEY AUTOINCREMENT, PublisherName VARCHAR(255) NOT NULL);

INSERT INTO Publisher VALUES(1,'Aniplex Inc.');

INSERT INTO Publisher VALUES(2,'Aniplex of America Inc.');

INSERT INTO Publisher VALUES(3,'Tsukihimates');

INSERT INTO Publisher VALUES(4,'Nemlei');

INSERT INTO Publisher VALUES(5,'Kit9 Studio');

INSERT INTO Publisher VALUES(6,'KeroQ');

INSERT INTO Publisher VALUES(7,'Frontwing');

INSERT INTO Publisher VALUES(8,'Denpasoft');

INSERT INTO Publisher VALUES(9,'Sekai Project');

INSERT INTO Publisher VALUES(10,'TYPE-MOON');

INSERT INTO Publisher VALUES(11,'Kadokawa Games');

INSERT INTO Publisher VALUES(12,'MangaGamer');

INSERT INTO Publisher VALUES(13,'NOVECT');

INSERT INTO Publisher VALUES(14,'Team Salvato');

INSERT INTO Publisher VALUES(15,'SBN3');

INSERT INTO Publisher VALUES(16,'PROTOTYPE');

INSERT INTO Publisher VALUES(17,'Key');

INSERT INTO Publisher VALUES(18,'Marvelous Inc.');

INSERT INTO Publisher VALUES(19,'Type-Moon Studio BB');

CREATE TABLE PublishedBy (NovelID INTEGER, PublisherID INTEGER, FOREIGN KEY (NovelID) REFERENCES Novel(NovelID), FOREIGN KEY (PublisherID) REFERENCES Publisher(PublisherID), PRIMARY KEY (NovelID, PublisherID));

INSERT INTO PublishedBy VALUES(1,12);

INSERT INTO PublishedBy VALUES(1,13);

INSERT INTO PublishedBy VALUES(2,16);

INSERT INTO PublishedBy VALUES(2,17);

INSERT INTO PublishedBy VALUES(3,15);

INSERT INTO PublishedBy VALUES(4,14);

INSERT INTO PublishedBy VALUES(5,18);

INSERT INTO PublishedBy VALUES(5,19);

INSERT INTO PublishedBy VALUES(6,10);

INSERT INTO PublishedBy VALUES(6,11);

INSERT INTO PublishedBy VALUES(7,1);

INSERT INTO PublishedBy VALUES(7,10);

INSERT INTO PublishedBy VALUES(7,11);

INSERT INTO PublishedBy VALUES(8,7);

INSERT INTO PublishedBy VALUES(8,8);

INSERT INTO PublishedBy VALUES(8,9);

INSERT INTO PublishedBy VALUES(8,16);

INSERT INTO PublishedBy VALUES(9,6);

INSERT INTO PublishedBy VALUES(9,7);

INSERT INTO PublishedBy VALUES(10,4);

INSERT INTO PublishedBy VALUES(10,5);

INSERT INTO PublishedBy VALUES(11,1);

INSERT INTO PublishedBy VALUES(11,2);

INSERT INTO PublishedBy VALUES(11,3);

CREATE TABLE WrittenBy (NovelID INTEGER, AuthorID INTEGER, FOREIGN KEY (NovelID) REFERENCES Novel(NovelID), FOREIGN KEY (AuthorID) REFERENCES Author(AuthorID), PRIMARY KEY (NovelID, AuthorID));

INSERT INTO WrittenBy VALUES(1,4);

INSERT INTO WrittenBy VALUES(2,1);

INSERT INTO WrittenBy VALUES(3,2);

INSERT INTO WrittenBy VALUES(4,3);

INSERT INTO WrittenBy VALUES(5,5);

INSERT INTO WrittenBy VALUES(5,10);

INSERT INTO WrittenBy VALUES(6,5);

INSERT INTO WrittenBy VALUES(7,5);

INSERT INTO WrittenBy VALUES(8,6);

INSERT INTO WrittenBy VALUES(9,8);

INSERT INTO WrittenBy VALUES(9,9);

INSERT INTO WrittenBy VALUES(10,7);

INSERT INTO WrittenBy VALUES(11,5);

CREATE TABLE VoicedBy (NovelID INTEGER, VoiceActorID INTEGER, FOREIGN KEY (NovelID) REFERENCES Novel(NovelID), FOREIGN KEY (VoiceActorID) REFERENCES VoiceActor(VoiceActorID), PRIMARY KEY (NovelID, VoiceActorID));

INSERT INTO VoicedBy VALUES(1,NULL);

INSERT INTO VoicedBy VALUES(2,NULL);

INSERT INTO VoicedBy VALUES(3,9);

INSERT INTO VoicedBy VALUES(3,10);

INSERT INTO VoicedBy VALUES(4,NULL);

INSERT INTO VoicedBy VALUES(5,11);

INSERT INTO VoicedBy VALUES(6,7);

INSERT INTO VoicedBy VALUES(7,8);

INSERT INTO VoicedBy VALUES(8,5);

INSERT INTO VoicedBy VALUES(9,3);

INSERT INTO VoicedBy VALUES(10,NULL);

INSERT INTO VoicedBy VALUES(11,1);

INSERT INTO VoicedBy VALUES(11,2);

INSERT INTO VoicedBy VALUES(9,4);

INSERT INTO VoicedBy VALUES(8,6);

INSERT INTO VoicedBy VALUES(7,7);

CREATE TABLE ComposedBy (NovelID INTEGER, ComposerID INTEGER, FOREIGN KEY (NovelID) REFERENCES Novel(NovelID), FOREIGN KEY (ComposerID) REFERENCES Composer(ComposerID), PRIMARY KEY (NovelID, ComposerID));

INSERT INTO ComposedBy VALUES(2,11);

INSERT INTO ComposedBy VALUES(3,NULL);

INSERT INTO ComposedBy VALUES(4,8);

INSERT INTO ComposedBy VALUES(5,1);

INSERT INTO ComposedBy VALUES(6,6);

INSERT INTO ComposedBy VALUES(7,1);

INSERT INTO ComposedBy VALUES(8,5);

INSERT INTO ComposedBy VALUES(9,4);

INSERT INTO ComposedBy VALUES(10,3);

INSERT INTO ComposedBy VALUES(11,2);

INSERT INTO ComposedBy VALUES(1,10);

CREATE TABLE BelongsToGenre (NovelID INTEGER, GenreID INTEGER, FOREIGN KEY (NovelID) REFERENCES Novel(NovelID), FOREIGN KEY (GenreID) REFERENCES Genre(GenreID), PRIMARY KEY (NovelID, GenreID));

INSERT INTO BelongsToGenre VALUES(1,1);

INSERT INTO BelongsToGenre VALUES(1,4);

INSERT INTO BelongsToGenre VALUES(1,6);

INSERT INTO BelongsToGenre VALUES(1,8);

INSERT INTO BelongsToGenre VALUES(1,9);

INSERT INTO BelongsToGenre VALUES(1,11);

INSERT INTO BelongsToGenre VALUES(1,12);

INSERT INTO BelongsToGenre VALUES(2,2);

INSERT INTO BelongsToGenre VALUES(2,6);

INSERT INTO BelongsToGenre VALUES(2,10);

INSERT INTO BelongsToGenre VALUES(2,13);

INSERT INTO BelongsToGenre VALUES(3,2);

INSERT INTO BelongsToGenre VALUES(3,5);

INSERT INTO BelongsToGenre VALUES(3,8);

INSERT INTO BelongsToGenre VALUES(3,10);

INSERT INTO BelongsToGenre VALUES(3,12);

INSERT INTO BelongsToGenre VALUES(3,14);

INSERT INTO BelongsToGenre VALUES(4,2);

INSERT INTO BelongsToGenre VALUES(4,3);

INSERT INTO BelongsToGenre VALUES(4,4);

INSERT INTO BelongsToGenre VALUES(4,6);

INSERT INTO BelongsToGenre VALUES(4,7);

INSERT INTO BelongsToGenre VALUES(4,8);

INSERT INTO BelongsToGenre VALUES(4,9);

INSERT INTO BelongsToGenre VALUES(4,13);

INSERT INTO BelongsToGenre VALUES(5,2);

INSERT INTO BelongsToGenre VALUES(5,5);

INSERT INTO BelongsToGenre VALUES(5,10);

INSERT INTO BelongsToGenre VALUES(5,12);

INSERT INTO BelongsToGenre VALUES(5,13);

INSERT INTO BelongsToGenre VALUES(6,2);

INSERT INTO BelongsToGenre VALUES(6,4);

INSERT INTO BelongsToGenre VALUES(6,5);

INSERT INTO BelongsToGenre VALUES(6,6);

INSERT INTO BelongsToGenre VALUES(6,8);

INSERT INTO BelongsToGenre VALUES(6,9);

INSERT INTO BelongsToGenre VALUES(6,11);

INSERT INTO BelongsToGenre VALUES(6,12);

INSERT INTO BelongsToGenre VALUES(6,13);

INSERT INTO BelongsToGenre VALUES(7,2);

INSERT INTO BelongsToGenre VALUES(7,4);

INSERT INTO BelongsToGenre VALUES(7,5);

INSERT INTO BelongsToGenre VALUES(7,6);

INSERT INTO BelongsToGenre VALUES(7,8);

INSERT INTO BelongsToGenre VALUES(7,10);

INSERT INTO BelongsToGenre VALUES(7,13);

INSERT INTO BelongsToGenre VALUES(8,2);

INSERT INTO BelongsToGenre VALUES(8,5);

INSERT INTO BelongsToGenre VALUES(8,6);

INSERT INTO BelongsToGenre VALUES(8,10);

INSERT INTO BelongsToGenre VALUES(8,13);

INSERT INTO BelongsToGenre VALUES(8,14);

INSERT INTO BelongsToGenre VALUES(9,2);

INSERT INTO BelongsToGenre VALUES(9,4);

INSERT INTO BelongsToGenre VALUES(9,5);

INSERT INTO BelongsToGenre VALUES(9,6);

INSERT INTO BelongsToGenre VALUES(9,7);

INSERT INTO BelongsToGenre VALUES(9,8);

INSERT INTO BelongsToGenre VALUES(9,10);

INSERT INTO BelongsToGenre VALUES(9,11);

INSERT INTO BelongsToGenre VALUES(9,12);

INSERT INTO BelongsToGenre VALUES(9,13);

INSERT INTO BelongsToGenre VALUES(10,2);

INSERT INTO BelongsToGenre VALUES(10,3);

INSERT INTO BelongsToGenre VALUES(10,4);

INSERT INTO BelongsToGenre VALUES(10,6);

INSERT INTO BelongsToGenre VALUES(10,7);

INSERT INTO BelongsToGenre VALUES(10,8);

INSERT INTO BelongsToGenre VALUES(10,9);

INSERT INTO BelongsToGenre VALUES(10,12);

INSERT INTO BelongsToGenre VALUES(10,13);

INSERT INTO BelongsToGenre VALUES(10,14);

INSERT INTO BelongsToGenre VALUES(11,2);

INSERT INTO BelongsToGenre VALUES(11,4);

INSERT INTO BelongsToGenre VALUES(11,5);

INSERT INTO BelongsToGenre VALUES(11,6);

INSERT INTO BelongsToGenre VALUES(11,7);

INSERT INTO BelongsToGenre VALUES(11,8);

INSERT INTO BelongsToGenre VALUES(11,9);

INSERT INTO BelongsToGenre VALUES(11,11);

INSERT INTO BelongsToGenre VALUES(11,13);

CREATE TABLE AvailableOn (NovelID INTEGER, PlatformID INTEGER, FOREIGN KEY (NovelID) REFERENCES Novel(NovelID), FOREIGN KEY (PlatformID) REFERENCES Platform(PlatformID), PRIMARY KEY (NovelID, PlatformID));

INSERT INTO AvailableOn VALUES(1,3);

INSERT INTO AvailableOn VALUES(1,7);

INSERT INTO AvailableOn VALUES(1,8);

INSERT INTO AvailableOn VALUES(1,12);

INSERT INTO AvailableOn VALUES(2,12);

INSERT INTO AvailableOn VALUES(2,8);

INSERT INTO AvailableOn VALUES(2,3);

INSERT INTO AvailableOn VALUES(2,4);

INSERT INTO AvailableOn VALUES(3,1);

INSERT INTO AvailableOn VALUES(3,2);

INSERT INTO AvailableOn VALUES(3,12);

INSERT INTO AvailableOn VALUES(3,13);

INSERT INTO AvailableOn VALUES(4,1);

INSERT INTO AvailableOn VALUES(4,3);

INSERT INTO AvailableOn VALUES(4,5);

INSERT INTO AvailableOn VALUES(4,6);

INSERT INTO AvailableOn VALUES(4,7);

INSERT INTO AvailableOn VALUES(4,8);

INSERT INTO AvailableOn VALUES(4,11);

INSERT INTO AvailableOn VALUES(4,12);

INSERT INTO AvailableOn VALUES(5,10);

INSERT INTO AvailableOn VALUES(6,11);

INSERT INTO AvailableOn VALUES(6,12);

INSERT INTO AvailableOn VALUES(7,11);

INSERT INTO AvailableOn VALUES(7,12);

INSERT INTO AvailableOn VALUES(7,13);

INSERT INTO AvailableOn VALUES(7,8);

INSERT INTO AvailableOn VALUES(8,12);

INSERT INTO AvailableOn VALUES(8,11);

INSERT INTO AvailableOn VALUES(8,8);

INSERT INTO AvailableOn VALUES(9,12);

INSERT INTO AvailableOn VALUES(10,12);

INSERT INTO AvailableOn VALUES(7,9);

INSERT INTO AvailableOn VALUES(11,3);

INSERT INTO AvailableOn VALUES(11,4);

INSERT INTO AvailableOn VALUES(11,8);

DELETE FROM sqlite\_sequence;

INSERT INTO sqlite\_sequence VALUES('Novel',11);

INSERT INTO sqlite\_sequence VALUES('Author',10);

INSERT INTO sqlite\_sequence VALUES('VoiceActor',11);

INSERT INTO sqlite\_sequence VALUES('Composer',11);

INSERT INTO sqlite\_sequence VALUES('Genre',14);

INSERT INTO sqlite\_sequence VALUES('Platform',13);

INSERT INTO sqlite\_sequence VALUES('Publisher',19);

COMMIT;

Java code:  
import java.sql.DriverManager;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.sql.PreparedStatement;

import java.util.\*;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.DataInputStream;

/\*\*

\*

\* @author sqlitetutorial.net

\*/

public class VisualNovelDatabase {

/\*\*

\* Connect to the test.db database

\* @return the Connection object

\*/

private Connection connect() {

// SQLite connection string

String url = "jdbc:sqlite:C://sqlite/vndb.db";

Connection conn = null;

try {

conn = DriverManager.getConnection(url);

} catch (SQLException e) {

System.out.println(e.getMessage());

}

return conn;

}

public void getTheBooksByPublisher(String publishername) { //case 1

String sql = "SELECT substr(novel.title||' ',1,10) title, IFNULL(novel.rating,0.0) rating, novel.agerating agerating, novel.releasedate releasedate FROM publisher " +

"INNER JOIN PublishedBy ON Publisher.PublisherID = PublishedBy.PublisherID " +

"inner join novel on PublishedBy.novelid = novel.novelid " +

"where publisher.publishername like ? COLLATE NOCASE";

try (Connection conn = this.connect();

PreparedStatement pstmt = conn.prepareStatement(sql)){

// set the value

pstmt.setString(1,publishername);

//

ResultSet rs = pstmt.executeQuery();

System.out.println("Title " + "\t\t\t" + "Rating" + "\t\t\t" + "Agerating" + "\t\t" + "Releasedate" );

System.out.println("------------------------------------------------------------------------------------------" );

// loop through the result set

while (rs.next()) {

System.out.println(rs.getString("title") + "\t\t\t" +

rs.getString("rating") + "\t\t\t" +

rs.getString("agerating") + "\t\t\t" +

rs.getString("releasedate") );

}

rs.close();

pstmt.close();

} catch (SQLException e) {

System.out.println(e.getMessage());

}

}

public void getTheBooksByAuthor(String authorname) { //case 2

String sql = "SELECT substr(novel.title||' ',1,10) title, IFNULL(novel.rating,0.0) rating, novel.agerating agerating, novel.releasedate releasedate "

+ "FROM author " +

"INNER JOIN WrittenBy ON author.AuthorId = writtenby.AuthorId " +

"inner join novel on WrittenBy.novelid = novel.novelid " +

"where author.authorname like ? COLLATE NOCASE";

try (Connection conn = this.connect();

PreparedStatement pstmt = conn.prepareStatement(sql)){

// set the value

pstmt.setString(1,authorname);

//

ResultSet rs = pstmt.executeQuery();

System.out.println("Title " + "\t\t\t" + "Rating" + "\t\t\t" + "Agerating" + "\t\t" + "Releasedate" );

System.out.println("------------------------------------------------------------------------------------------" );

// loop through the result set

while (rs.next()) {

System.out.println(rs.getString("title") + "\t\t\t" +

rs.getString("rating") + "\t\t\t" +

rs.getString("agerating") + "\t\t\t" +

rs.getString("releasedate") );

}

rs.close();

pstmt.close();

} catch (SQLException e) {

System.out.println(e.getMessage());

}

}

public void getLeadingMaleVoiceActor(String novelTitle) { //case 3

String sql = "SELECT VoiceActor.VoiceName " +

"FROM VoiceActor " +

"INNER JOIN VoicedBy ON VoiceActor.VoiceActorID = VoicedBy.VoiceActorID " +

"INNER JOIN Novel ON VoicedBy.NovelID = Novel.NovelID " +

"WHERE Novel.Title = ? AND VoiceActor.Gender = 'M' " +

"LIMIT 1";

try (Connection conn = this.connect();

PreparedStatement pstmt = conn.prepareStatement(sql)) {

// set the value

pstmt.setString(1, novelTitle);

ResultSet rs = pstmt.executeQuery();

// Check if there is a result

if (rs.next()) {

System.out.println("The leading male voice actor of the visual novel \"" + novelTitle + "\" is: " +

rs.getString("VoiceName"));

} else {

System.out.println("No information found for the leading male voice actor of the visual novel \"" + novelTitle + "\".");

}

rs.close();

pstmt.close();

} catch (SQLException e) {

System.out.println(e.getMessage());

}

}

public void getLeadingFemaleVoiceActress(String novelTitle) { //case 4

String sql = "SELECT VoiceActor.VoiceName " +

"FROM VoiceActor " +

"INNER JOIN VoicedBy ON VoiceActor.VoiceActorID = VoicedBy.VoiceActorID " +

"INNER JOIN Novel ON VoicedBy.NovelID = Novel.NovelID " +

"WHERE Novel.Title = ? AND VoiceActor.Gender = 'F' " +

"LIMIT 1";

try (Connection conn = this.connect();

PreparedStatement pstmt = conn.prepareStatement(sql)) {

// set the value

pstmt.setString(1, novelTitle);

ResultSet rs = pstmt.executeQuery();

// Check if there is a result

if (rs.next()) {

System.out.println("The leading female voice actress of the visual novel \"" + novelTitle + "\" is: " +

rs.getString("VoiceName"));

} else {

System.out.println("No information found for the leading female voice actress of the visual novel \"" + novelTitle + "\".");

}

rs.close();

pstmt.close();

} catch (SQLException e) {

System.out.println(e.getMessage());

}

}

public void getAgeRatingForNovel(String novelTitle) { //case 5

String sql = "SELECT AgeRating " +

"FROM Novel " +

"WHERE Title = ?";

try (Connection conn = this.connect();

PreparedStatement pstmt = conn.prepareStatement(sql)) {

// set the value

pstmt.setString(1, novelTitle);

ResultSet rs = pstmt.executeQuery();

// Check if there is a result

if (rs.next()) {

System.out.println("The age rating for the reading of the visual novel \"" + novelTitle + "\" is: " +

rs.getString("AgeRating"));

} else {

System.out.println("No information found for the age rating of the visual novel \"" + novelTitle + "\".");

}

rs.close();

pstmt.close();

} catch (SQLException e) {

System.out.println(e.getMessage());

}

}

public void getLowestRatingNovel() {

String sql = "SELECT Novel.Title, MIN(Novel.Rating) AS LowestRating " +

"FROM Novel " +

"WHERE Novel.Rating IS NOT NULL";

try (Connection conn = this.connect();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql)) {

// Check if there is a result

if (rs.next()) {

String title = rs.getString("Title");

double lowestRating = rs.getDouble("LowestRating");

System.out.println("The visual novel with the lowest rating is: " + title + " with a rating of " + lowestRating);

} else {

System.out.println("No information found for the visual novel with the lowest rating.");

}

} catch (SQLException e) {

System.out.println(e.getMessage());

}

}

public void getMostCommonGenres() { //case 7

String sql = "SELECT Genre.GenreName, COUNT(\*) AS NovelCount, GROUP\_CONCAT(Novel.Title, ', ') AS Titles " +

"FROM Genre " +

"INNER JOIN BelongsToGenre ON Genre.GenreID = BelongsToGenre.GenreID " +

"INNER JOIN Novel ON BelongsToGenre.NovelID = Novel.NovelID " +

"GROUP BY Genre.GenreName " +

"ORDER BY NovelCount DESC";

try (Connection conn = this.connect();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql)) {

System.out.println("Most common genres:");

System.out.println("Genre Name \tNovel Count\t\t\tTitles");

// Loop through the result set

while (rs.next()) {

String genreName = rs.getString("GenreName");

int novelCount = rs.getInt("NovelCount");

String titles = rs.getString("Titles");

System.out.println(genreName + "\t\t" + novelCount + "\t\t\t\t" + titles);

}

} catch (SQLException e) {

System.out.println(e.getMessage());

}

}

public void getReleaseDate(String novelTitle) { //case 8

String sql = "SELECT ReleaseDate " +

"FROM Novel " +

"WHERE Title LIKE ? COLLATE NOCASE";

try (Connection conn = this.connect();

PreparedStatement pstmt = conn.prepareStatement(sql)) {

// Set the value for the prepared statement

pstmt.setString(1, novelTitle);

ResultSet rs = pstmt.executeQuery();

// Check if there is a result

if (rs.next()) {

String releaseDate = rs.getString("ReleaseDate");

System.out.println("The visual novel '" + novelTitle + "' was released on: " + releaseDate);

} else {

System.out.println("No information found for the visual novel '" + novelTitle + "'.");

}

rs.close();

pstmt.close();

} catch (SQLException e) {

System.out.println(e.getMessage());

}

}

public void getReadingStatus(String novelTitle) { //case 9

String sql = "SELECT ReadingStatus " +

"FROM Novel " +

"WHERE Title LIKE ? COLLATE NOCASE";

try (Connection conn = this.connect();

PreparedStatement pstmt = conn.prepareStatement(sql)) {

// Set the value for the prepared statement

pstmt.setString(1, novelTitle);

ResultSet rs = pstmt.executeQuery();

// Check if there is a result

if (rs.next()) {

String readingStatus = rs.getString("ReadingStatus");

System.out.println("The reading status of the novel '" + novelTitle + "' is: " + readingStatus);

} else {

System.out.println("No information found for the novel '" + novelTitle + "'.");

}

rs.close();

pstmt.close();

} catch (SQLException e) {

System.out.println(e.getMessage());

}

}

public void getMainComposer(String novelTitle) { //case 10

String sql = "SELECT Composer.ComposerName " +

"FROM Composer " +

"INNER JOIN ComposedBy ON Composer.ComposerID = ComposedBy.ComposerID " +

"INNER JOIN Novel ON ComposedBy.NovelID = Novel.NovelID " +

"WHERE Novel.Title LIKE ? COLLATE NOCASE";

try (Connection conn = this.connect();

PreparedStatement pstmt = conn.prepareStatement(sql)) {

// Set the value for the prepared statement

pstmt.setString(1, novelTitle);

ResultSet rs = pstmt.executeQuery();

// Check if there is a result

if (rs.next()) {

String mainComposer = rs.getString("ComposerName");

System.out.println("The main music composer of the novel '" + novelTitle + "' is: " + mainComposer);

} else {

System.out.println("No information found for the novel '" + novelTitle + "'.");

}

rs.close();

pstmt.close();

} catch (SQLException e) {

System.out.println(e.getMessage());

}

}

public void getAverageCompletionTime() { //case 11

String sql = "SELECT AVG(TimeToFinish) AS AvgCompletionTime " +

"FROM Novel " +

"WHERE ReadingStatus = 'Completed'";

try (Connection conn = this.connect();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql)) {

// Check if there is a result

if (rs.next()) {

double avgCompletionTime = rs.getDouble("AvgCompletionTime");

System.out.println("The average completion time of visual novels is: " + avgCompletionTime + " hours");

} else {

System.out.println("No information found for the average completion time of visual novels.");

}

} catch (SQLException e) {

System.out.println(e.getMessage());

}

}

public static void main(String[] args) {

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("\* Welcome to the visual novel database application \*");

System.out.println("\* \*");

System.out.println("\* 01. Books by publisher \*");

System.out.println("\* 02. Books by writer (first and last name) \*");

System.out.println("\* 03. What is the name of the leading male voice actor of the visual novel? \*");

System.out.println("\* 04. What is the name of the leading female voice actress of the visual novel? \*");

System.out.println("\* 05. What is the given age rating for the reading? \*");

System.out.println("\* 06. What was the lowest rating I gave a visual novel? \*");

System.out.println("\* 07. What genres were most common? \*");

System.out.println("\* 08. When was the visual novel released? \*");

System.out.println("\* 09. Have I read the novel, am I currently reading it, or is it on my wish list (or various such statuses)? \*");

System.out.println("\* 10. Who was the main music composer of the visual novel? \*");

System.out.println("\* 11. How long does the visual novel take to complete, on average? \*");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Scanner choice = new Scanner(System.in);

System.out.print("Enter your choice: ");

int n = choice.nextInt();

switch (n) {

case 1:

VisualNovelDatabase app1 = new VisualNovelDatabase();

InputStreamReader inStream = new InputStreamReader(System.in);

BufferedReader stdind = new BufferedReader(inStream);

String str[] = new String[1];

try {

System.out.print("Enter publisher: ");

str[0] = stdind.readLine();

//System.out.println("Enter it:");

//str[1] = stdind.readLine();

} catch (IOException ioe) { ioe.printStackTrace();

}

//choice.close();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

app1.getTheBooksByPublisher("%"+str[0]+"%");

System.out.println();

System.out.println();

//app1.getThePublisher("Nemlei");

break;

case 2:

VisualNovelDatabase app2 = new VisualNovelDatabase();

InputStreamReader inStream2 = new InputStreamReader(System.in);

BufferedReader stdind2 = new BufferedReader(inStream2);

String str2[] = new String[1];

try {

System.out.print("Enter author: ");

str2[0] = stdind2.readLine();

//System.out.println("Enter it:");

//str[1] = stdind.readLine();

} catch (IOException ioe) { ioe.printStackTrace();

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

app2.getTheBooksByAuthor("%"+str2[0]+"%");

System.out.println();

System.out.println();

break;

case 3:

VisualNovelDatabase app3 = new VisualNovelDatabase();

InputStreamReader inStream3 = new InputStreamReader(System.in);

BufferedReader stdind3 = new BufferedReader(inStream3);

String novelTitle = "";

try {

System.out.print("Enter the title of the visual novel: ");

novelTitle = stdind3.readLine();

} catch (IOException ioe) {

ioe.printStackTrace();

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

app3.getLeadingMaleVoiceActor(novelTitle);

System.out.println();

System.out.println();

break;

case 4:

VisualNovelDatabase app4 = new VisualNovelDatabase();

InputStreamReader inStream4 = new InputStreamReader(System.in);

BufferedReader stdind4 = new BufferedReader(inStream4);

String novelTitle4 = "";

try {

System.out.print("Enter the title of the visual novel: ");

novelTitle4 = stdind4.readLine();

} catch (IOException ioe) {

ioe.printStackTrace();

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

app4.getLeadingFemaleVoiceActress(novelTitle4);

System.out.println();

System.out.println();

break;

case 5:

VisualNovelDatabase app5 = new VisualNovelDatabase();

InputStreamReader inStream5 = new InputStreamReader(System.in);

BufferedReader stdind5 = new BufferedReader(inStream5);

String novelTitle5 = "";

try {

System.out.print("Enter the title of the visual novel: ");

novelTitle5 = stdind5.readLine();

} catch (IOException ioe) {

ioe.printStackTrace();

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

app5.getAgeRatingForNovel(novelTitle5);

System.out.println();

System.out.println();

break;

case 6:

VisualNovelDatabase app6 = new VisualNovelDatabase();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

app6.getLowestRatingNovel();

System.out.println();

System.out.println();

break;

case 7:

VisualNovelDatabase app7 = new VisualNovelDatabase();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

app7.getMostCommonGenres();

System.out.println();

System.out.println();

break;

case 8:

VisualNovelDatabase app8 = new VisualNovelDatabase();

InputStreamReader inStream8 = new InputStreamReader(System.in);

BufferedReader stdind8 = new BufferedReader(inStream8);

try {

System.out.print("Enter the title of the visual novel: ");

String inputTitle = stdind8.readLine();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

app8.getReleaseDate(inputTitle);

System.out.println();

System.out.println();

} catch (IOException e) {

e.printStackTrace();

}

break;

case 9:

VisualNovelDatabase app9 = new VisualNovelDatabase();

InputStreamReader inStream9 = new InputStreamReader(System.in);

BufferedReader stdind9 = new BufferedReader(inStream9);

try {

System.out.print("Enter the title of the novel: ");

String inputTitle = stdind9.readLine();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

app9.getReadingStatus(inputTitle);

System.out.println();

System.out.println();

} catch (IOException e) {

e.printStackTrace();

}

break;

case 10:

VisualNovelDatabase app10 = new VisualNovelDatabase();

InputStreamReader inStream10 = new InputStreamReader(System.in);

BufferedReader stdind10 = new BufferedReader(inStream10);

try {

System.out.print("Enter the title of the novel: ");

String inputTitle = stdind10.readLine();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

app10.getMainComposer(inputTitle);

System.out.println();

System.out.println();

} catch (IOException e) {

e.printStackTrace();

}

break;

case 11:

VisualNovelDatabase app11 = new VisualNovelDatabase();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

app11.getAverageCompletionTime();

System.out.println();

System.out.println();

break;

default:

System.out.println("Invalid choice. Please enter a number between 1 and 12.");

break;

} // end switch statement

choice.close();

}

}