```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data.SQLite;
using Microsoft.Xna.Framework.Graphics;
using System.Threading;
using Microsoft.Xna.Framework;

namespace coursework
{
    class DBManager
    {
        public static List<string> commands = new List<string>();
        /// <summary>
        /// connect to the database
        /// </summary>
        /// <returns></returns>
        public static SQLiteConnection CreateConnection()
        {

            SQLiteConnection sqlite_conn;
            // Create a new database connection:
            sqlite_conn = new SQLiteConnection("Data Source= Database.db; Version = 3; New = True; Compress = True; ");
            // Open the connection:
            try
            {
                sqlite_conn.Open();
            }
            catch
            {

            }
            return sqlite_conn;
        }

        /// <summary>
        /// if the command queue is not empty, calls RunCommand
        /// if command queue is empty, thread sleeps until a new command is added
        /// </summary>
        public static void commandQueue()
        {
            while (Game1.GameActive)
            {
                if (commands.Count != 0)
                {
                    RunCommand(commands[0]);
                    commands.RemoveAt(0);
                }
                else
                {
                    try
                    {
                        Thread.Sleep(Timeout.Infinite);
                    }
                    catch
                    {
                        //sleep interrupted
                    }
                }
            }
        }

        /// <summary>
        /// when database is unlocked first command in commands list is run
        /// </summary>
        /// <param name="cmd"></param>
        public static void RunCommand(string cmd)
        {
            SQLiteConnection conn;
            SQLiteCommand sqlite_cmd;
            conn = CreateConnection();
            sqlite_cmd = conn.CreateCommand();
            sqlite_cmd.CommandText = cmd;
            WaitForDbToBeUnlocked();
            try
            {
                sqlite_cmd.ExecuteNonQuery();
                conn.Close();
            }
            catch
            {

            }
        }

        /// <summary>
        /// returns true if databse is locked
        /// </summary>
        /// <returns></returns>
        public static bool IsDatabaseLocked()
        {
            bool locked = true;
            CreateConnection();
            SQLiteConnection connection = new SQLiteConnection($"Data Source=Database.db;Version=3;");
            connection.Open();

            try
            {
                SQLiteCommand beginCommand = connection.CreateCommand();
                beginCommand.CommandText = "BEGIN EXCLUSIVE"; // tries to acquire the lock
                                                              // CommandTimeout is set to 0 to get error immediately if DB is locked
                                                              // otherwise it will wait for 30 sec by default
                beginCommand.CommandTimeout = 0;
                beginCommand.ExecuteNonQuery();

                SQLiteCommand commitCommand = connection.CreateCommand();
                commitCommand.CommandText = "COMMIT"; // releases the lock immediately
                commitCommand.ExecuteNonQuery();
                locked = false;
            }
            catch (SQLiteException)
            {
                // database is locked error
            }
            finally
            {
                connection.Close();
            }

            return locked;
        }

        /// <summary>
        /// calls IsDatabaseLocked every 1 millisecond until it returns false
        /// </summary>
        /// <returns></returns>
        public static bool WaitForDbToBeUnlocked()
        {
            while (IsDatabaseLocked())
            {
                Thread.Sleep(1);
            }
            return true;
        }

        /// <summary>
        /// creates database with starting data set
        /// </summary>
        /// <param name="conn"></param>
        public static void InitialiseDB(SQLiteConnection conn)
        {

            SQLiteCommand sqlite_cmd;
            string CreateChar = "CREATE TABLE Characters (CharacterID INTEGER, Name VARCHAR(20), Level INT, Race VARCHAR(20), Dexterity INTEGER, Strength INTEGER, ArmorClass INTEGER, UserID INTEGER, PRIMARY KEY(CharacterID AUTOINCREMENT))";
            string CreateWeapons = "CREATE TABLE Weapons (WeaponName VARCHAR(20), Range INT, hitDie INT)";
            string CreateMonsters = "CREATE TABLE Monsters (MonsterID INTEGER, MonsterName VARCHAR(20), ArmorClass INTEGER, Strength INTEGER, Dex INTEGER, HitPoints INTEGER, HitDie INTEGER, Range INTEGER, PRIMARY KEY(MonsterID AUTOINCREMENT))";
            string CreateRace = "CREATE TABLE Race (RaceName VARCHAR(20), DexModifier INTEGER, StrengthModifier INTEGER, Speed INTEGER, PRIMARY KEY(RaceName))";
            string CreateWeaponAssignment = "CREATE TABLE MonsterWeaponAssignment (MonsterID INTEGER, WeaponID INTEGER, PRIMARY KEY(WeaponID, MonsterID))";

            sqlite_cmd = conn.CreateCommand();
            sqlite_cmd.CommandText = CreateChar;
            sqlite_cmd.ExecuteNonQuery();
            sqlite_cmd.CommandText = CreateWeapons;
            sqlite_cmd.ExecuteNonQuery();
            sqlite_cmd.CommandText = CreateMonsters;
            sqlite_cmd.ExecuteNonQuery();
            sqlite_cmd.CommandText = CreateRace;
            sqlite_cmd.ExecuteNonQuery();
            sqlite_cmd.CommandText = CreateWeaponAssignment;
            sqlite_cmd.ExecuteNonQuery();

            sqlite_cmd.CommandText = "INSERT INTO Weapons (WeaponName, Range, hitDie) VALUES('dagger', 20 , 4);";
            sqlite_cmd.ExecuteNonQuery();
            sqlite_cmd.CommandText = "INSERT INTO Weapons (WeaponName, Range, hitDie) VALUES('longbow', 150 , 8);";
            sqlite_cmd.ExecuteNonQuery();
            sqlite_cmd.CommandText = "INSERT INTO Weapons (WeaponName, Range, hitDie) VALUES('spear', 20 , 6);";
            sqlite_cmd.ExecuteNonQuery();

            sqlite_cmd.CommandText = "INSERT INTO Monsters (MonsterName, ArmorClass, Strength, Dex, HitPoints, HitDie, Range) VALUES('Awakened Shrub', 9, 3, 8, 10, 4, 9);";
            sqlite_cmd.ExecuteNonQuery();
            sqlite_cmd.CommandText = "INSERT INTO Monsters (MonsterName, ArmorClass, Strength, Dex, HitPoints) VALUES('Goblin', 15, 8, 14, 7);";
            sqlite_cmd.ExecuteNonQuery();

            sqlite_cmd.CommandText = "INSERT INTO MonsterWeaponAssignment (MonsterID, WeaponID) VALUES( 2, 5);";
            sqlite_cmd.ExecuteNonQuery();
            sqlite_cmd.CommandText = "INSERT INTO MonsterWeaponAssignment (MonsterID, WeaponID) VALUES( 2, 4);";
            sqlite_cmd.ExecuteNonQuery();

            sqlite_cmd.CommandText = "INSERT INTO Race (RaceName, DexModifier, StrengthModifier, Speed) VALUES('Human', 1, 1, 30);";
            sqlite_cmd.ExecuteNonQuery();
            sqlite_cmd.CommandText = "INSERT INTO Race (RaceName, DexModifier, StrengthModifier, Speed) VALUES('Elf', 2, 0, 30);";
            sqlite_cmd.ExecuteNonQuery();
            sqlite_cmd.CommandText = "INSERT INTO Race (RaceName, DexModifier, StrengthModifier, Speed) VALUES('Dwarf', 0, 2, 25);";
            sqlite_cmd.ExecuteNonQuery();
        }

        /// <summary>
        /// creates insert SQL command for a character
        /// adds command to commands list
        /// </summary>
        /// <param name="name"></param>
        /// <param name="race"></param>
        /// <param name="level"></param>
        /// <param name="dex"></param>
        /// <param name="str"></param>
        /// <param name="AC"></param>
        /// <param name="userID"></param>
        public static void InsertData(string name, string race, int level, string dex, string str, int AC, string userID)
        {
            string cmd = "INSERT INTO Characters (Name, Level, Race, Dexterity, Strength, ArmorClass, UserID) VALUES('" + name + "', " + level + ", '" + race + "', " + dex + ", " + str + ", " + AC + ", '" + userID + "'); ";
            commands.Add(cmd);
        }

        #region read methods
        /// <summary>
        /// runs read command and returns resulting data
        /// </summary>
        /// <param name="cmd"></param>
        /// <returns></returns>
        public static SQLiteDataReader ExecuteReader(string cmd)
        {
            SQLiteDataReader data;
            SQLiteConnection conn;
            SQLiteCommand sqlite_cmd;
            conn = CreateConnection();
            sqlite_cmd = conn.CreateCommand();
            sqlite_cmd.CommandText = cmd;
            data = sqlite_cmd.ExecuteReader();
            return data;
        }

        /// <summary>
        /// reads character information from data base where the username matches the users input
        /// outputs the results
        /// </summary>
        /// <param name="conn"></param>
        /// <param name="spriteBatch"></param>
        /// <param name="font"></param>
        /// <param name="username"></param>
        public static void ReadCharacters(SQLiteConnection conn, SpriteBatch spriteBatch, SpriteFont font, string username)
        {
            int count = 0;
            SQLiteDataReader sqlite_datareader;
            string cmd = "SELECT * FROM Characters WHERE Characters.UserID = '" + username + "'";
            int height = 100;

            sqlite_datareader = DBManager.ExecuteReader(cmd);


            spriteBatch.DrawString(font, "ID", new Vector2(50, 50), Color.Pink);

            string col2 = sqlite_datareader.GetName(1);
            spriteBatch.DrawString(font, col2, new Vector2(125, 50), Color.Pink);

            string col3 = sqlite_datareader.GetName(2);
            spriteBatch.DrawString(font, col3, new Vector2(250, 50), Color.Pink);

            string col4 = sqlite_datareader.GetName(3);
            spriteBatch.DrawString(font, col4, new Vector2(350, 50), Color.Pink);

            string col5 = sqlite_datareader.GetName(4);
            spriteBatch.DrawString(font, "Dex", new Vector2(450, 50), Color.Pink);

            string col6 = sqlite_datareader.GetName(5);
            spriteBatch.DrawString(font, "Str", new Vector2(550, 50), Color.Pink);

            string col7 = sqlite_datareader.GetName(6);
            spriteBatch.DrawString(font, "AC", new Vector2(650, 50), Color.Pink);


            while (sqlite_datareader.Read()) // closed too soon
            {
                if (count - (6 * Game1.pageNum) < 6 && count - (6 * Game1.pageNum) >= 0)
                {
                    string myreader0 = sqlite_datareader.GetInt32(0).ToString();
                    string myreader1 = sqlite_datareader.GetString(1);
                    string myreader2 = sqlite_datareader.GetInt32(2).ToString();
                    string myreader3 = sqlite_datareader.GetString(3);
                    string myreader4 = sqlite_datareader.GetInt32(4).ToString();
                    string myreader5 = sqlite_datareader.GetInt32(5).ToString();
                    string myreader6 = sqlite_datareader.GetInt32(6).ToString();

                    spriteBatch.DrawString(font, myreader0, new Vector2(50, height), Color.Pink);
                    spriteBatch.DrawString(font, myreader1, new Vector2(125, height), Color.Pink);
                    spriteBatch.DrawString(font, myreader2, new Vector2(250, height), Color.Pink);
                    spriteBatch.DrawString(font, myreader3, new Vector2(350, height), Color.Pink);
                    spriteBatch.DrawString(font, myreader4, new Vector2(450, height), Color.Pink);
                    spriteBatch.DrawString(font, myreader5, new Vector2(550, height), Color.Pink);
                    spriteBatch.DrawString(font, myreader6, new Vector2(650,
```

```csharp
                    height), Color.Pink);
                    height += 50;
                }
                count++;
            }
        }

        /// <summary>
        /// reads all weapon data from the database and outputs it
        /// </summary>
        /// <param name="conn"></param>
        /// <param name="spriteBatch"></param>
        /// <param name="font"></param>
        public static void ReadWeapons(SQLiteConnection conn, SpriteBatch spriteBatch, SpriteFont font)
        {
            int count = 0;
            SQLiteDataReader sqlite_datareader;
            SQLiteCommand sqlite_cmd;
            sqlite_cmd = conn.CreateCommand();
            sqlite_cmd.CommandText = "SELECT * FROM Weapons";
            int height = 100;
            sqlite_datareader = sqlite_cmd.ExecuteReader();


            spriteBatch.DrawString(font, "ID", new Vector2(50, 50), Color.Pink);

            string col2 = sqlite_datareader.GetName(1);
            spriteBatch.DrawString(font, col2, new Vector2(125, 50), Color.Pink);

            string col3 = sqlite_datareader.GetName(2);
            spriteBatch.DrawString(font, col3, new Vector2(250, 50), Color.Pink);

            string col4 = sqlite_datareader.GetName(3);
            spriteBatch.DrawString(font, col4, new Vector2(350, 50), Color.Pink);


            while (sqlite_datareader.Read())
            {
                if (count - (6 * Game1.pageNum) < 6 && count - (6 * Game1.pageNum) >= 0)
                {
                    string myreader0 = sqlite_datareader.GetInt32(0).ToString();
                    string myreader1 = sqlite_datareader.GetString(1);
                    string myreader2 = sqlite_datareader.GetInt32(2).ToString();
                    string myreader3 = sqlite_datareader.GetInt32(3).ToString();

                    spriteBatch.DrawString(font, myreader0, new Vector2(50, height), Color.Pink);
                    spriteBatch.DrawString(font, myreader1, new Vector2(125, height), Color.Pink);
                    spriteBatch.DrawString(font, myreader2, new Vector2(250, height), Color.Pink);
                    spriteBatch.DrawString(font, myreader3, new Vector2(350, height), Color.Pink);
                    height += 50;
                }
                count++;
            }
        }

        public static void ReadModifiers(SQLiteConnection conn, ref int dexMod, ref int strMod, string race, ref int speed)
        {
            SQLiteDataReader sqlite_datareader;
            SQLiteCommand sqlite_cmd;
            sqlite_cmd = conn.CreateCommand();
            sqlite_cmd.CommandText = "SELECT DexModifier, StrengthModifier, Speed FROM Race WHERE RaceName = '" + race + "'";
            sqlite_datareader = sqlite_cmd.ExecuteReader();
            if (sqlite_datareader.Read())
            {
                dexMod = sqlite_datareader.GetInt32(0);
                strMod = sqlite_datareader.GetInt32(1);
                speed = sqlite_datareader.GetInt32(2);
            }
        }
        #endregion
    }
}
```