Project 1

Project Info

Due Date

End of Week 3: Saturday Sept. 21st

LATE SUBMISSIONS WILL NOT BE ACCEPTED

Submission: Submit to the Project 1 Assignments submissions folder/drop box.

See "Submission Requirements" below for details on how to generate the required file.

This is a group (max 2 students) assignment & must be done without using cognitive AI tools (like ChatGPT). Significant overlap between submissions from two or more students may be flagged for plagiarism.

Grading

Item	Description	Weight
Tables	**- All required entities are represented** **- Any tables required to model many to many relationships are present**	**20%**
Columns	**- All required attributes are represented, including columns required by technical requirements** **- All datatypes are fairly reasonable for the data they will store** **- Column nullability is appropriate (i.e. optional versus required)**	**20%**
Relationships	**- All required foreign key relationships are present** **- All foreign keys have the correct parent and child table (i.e. primary and foreign table, respectively)**	**20%**
Constraints	- All required default constraints and identities are present.	**10%**
Diagram	- Logical data model must be included. Should be a Crow's Foot (from Visio) or SQL Server diagram. SQL Server diagram is preferred. Images (JPG) are acceptable, provided they are clear and legible & show all tables, fields & relationships.	**10%**
Normalization	**- Proposed design is correct 3NF**	**20%**
TOTAL		**100%**

Deductions:

Submission Requirements	Penalty for failure to adhere to the submission requirements.	Up to - 10%
Error on Execution	**Penalty for failure to run without modification or error. You may assume that I will be using the correct database (i.e. "DROP/CREATE/USE DATABASE FanshaweDroneShare") is encouraged but not required (This way you can re-run YOUR solution many times AS you build it – the Ideal approach) - I can add it if it is missing.**	**Up to - 20%**

Business Requirements

You are working for Fanshawe College who are setting up a drone (small personal unmanned flying aircraft) sharing program and database called FanshaweDroneShare. They would like to track the people borrowing/renting, and the specific drones and accessories in the program.

Drones and accessories like cameras, GPS, sensors, joysticks are kept at stations (often the local library branch but not always). Each drone/accessory has a home station that Fanshawe employees will return it to occasionally (note there is no need to model/capture this work). The system will also track the station the drones/accessories are currently at. The current station will only be changed when a drone/accessory is checked in, so the current station for a drone/accessory will never be unknown. Note that Fanshawe may want to add other types of accessories in the future.

Stations have names and maximum number of drones that can be held, each of which are always stored. For each station, the system should be able to track the number of drones that are currently at the terminal. Drones will always have identification markings regulated by Transport Canada, and drones and accessories have manufacturer name, weight (in grams), model names and serial numbers which are always available. Some drones/accessories will also have a manufactured date (and some will not).

Pilots will set up accounts and will be charged for their use via those accounts. Accounts may cover more than one pilot, such as when a house of roommates sets up an account. Each pilot may also be associated with more than one account. each account could have more than one pilot and vice versa

When a pilot checks out a drone/accessory, it will be kept track of in the system. A pilot is permitted to sign out multiple drones/accessories at the same time. For example, a single pilot may sign out a drone for personal use as well as a drone for a guest. One drone/accessory will never be checked out to multiple pilots simultaneously.
each pilot could sign out more than one drone or accessory

The system will be used to store specific information when pilots open an account. It will need to track a pilot's first name and last name, along with their Transport Canada drone pilot certificate number, SIN and date of birth. We also need to store the street address, city, province, and postal code for a pilot. It is possible for multiple pilots to live at the same address (e.g. multiple pilots in the same house). It is also possible for one pilot to have multiple addresses in our system (e.g. home address, business address). The pilot's name, SIN, drone pilot certificate and date of birth are all mandatory, but all other pilot information is optional. we need constraints here for some attributes

For each account the opening date, current balance, and account number should be stored. The account number is a unique number created by another system at a bank, so will always be available. The opening date and current balance will also always be populated. we have constraints here

Technical Requirements

In addition to satisfying the business requirements, you have been asked to follow these technical standards.

1) A database diagram (ideally from SSMS) of your logical model for this system must be submitted.
2) There should be an identity column on every table in the database named <tablename>ID (e.g. a table named "MyTable" would have an ID called "MyTableID"). This should be implemented as an identity (i.e. auto-incrementing column).
3) All columns that are described as mandatory should not be nullable.
4) Ensure that all related tables are properly constrained using foreign keys.
5) This schema should be created in a new database called "FanshaweDroneShare"
6) All foreign key columns should have the same name as the column they reference.
7) The nullability of all foreign key columns should match the cardinality of the relationship they implement. I.e. "zero or one" is optional, whereas "exactly one" is not.
8) When there is more than one relationship between two entities, foreign columns should have descriptions added as prefixes to differentiate them.
9) Junction tables should be named by combining the names of the two tables joined. For example, a junction between TableA and TableB would be TableATableB.

10) Only the attributes/fields explicitly included or mentioned in the requirements should be included in the design. Do not add any columns that are not specifically asked for in these requirements.
11) The database created to satisfy these requirements should be properly normalized.

Submission Requirements

You will submit a database script created & run in SQL Server. Ideally you will submit the raw SQL code you used to create your database, but you can use SSMS to create the generated script after finishing the design using the GUI design tools in SSMS if you would like (note as I mentioned in class this is not the best way to learn/ensure you are following the design best practises, since every DBMS/IDE will have different GUI tools that won't usually ensure best design). **Make sure you are satisfied with the database you have created before scripting it out.**

1st Submission File (.txt): Continue using the Project 1 Sample SQL script and

1. Add your own scripts for the new required tables into the same file. Use comments to separate the answers for each requirement
2. Once you finish, rename the SQL file name, then submit it.

Note: Ideally you will create the database directly using the SQL code required to create the tables/relationships/constraints (vs using the GUI design tool/wizards – as mentioned in class using these GUI design tools/wizards is not the best way to learn SQL/design since these tools will always be different from one DBMS to another, but the SQL will be much more standard/portable). If you have the complete SQL code, including all the CREATE TABLE, Relationships/ Constraints, etc. as per the requirements above then submit all that SQL code in 1 .sql file (and in a separate version you save with a .txt subscript).

2nd Submission File (.sql): Script out the database:

1) Right click on your database and select "Tasks->Generate Scripts…"
2) If you see an introduction page, click "Next >"
3) On the "Choose Objects" page, select "Select specific database objects".
4) Ensure that all the tables you wish to submit are checked off.
5) Click "Next >"
6) On the "Set Scripting Options" page, ensure that "Save scripts to a specific location", "Save to file", and "Single file" are all selected.
7) Change the filename so that it is your **first and last name** with a .sql extension (e.g. "BruceWayne.sql")
8) Click "Next >"
9) Review your selections and click "Next >"

This will create a .sql file. Submit this file to the appropriate assignments dropbox.

You must test your script to ensure it executes in SQL Server before submitting - from scratch – ie. No FanshaweDroneShare Database created yet - so the first few SQL DDL build statements should include these statements :

```
Use Master;
GO
Alter database FanshaweDroneShare set single_user with rollback immediate;
GO
DROP Database FanshaweDroneShare;
GO
CREATE DATABASE FanshaweDroneShare;
GO
USE FanshaweDroneShare;
GO
```