CSE 581

Project 2

Project 2 task:

In project 1 you designed a database. In project 2 you will implement your design, load data into it and create other database objects.

Requirements:

1. Design – you are expected to make design adjustments[1] based on my feedback and issues you identify while working on project 2. You are required to submit a brief[2] list of the modifications. If your design was perfect, and you didn't need to make a single change, please state that.
2. Table creation – implement all of the tables from your design. Make sure you use identities, correct data types and all applicable constraints. You will submit all SQL used, as text. No screenshots.
3. Data load – load test data into your tables. Most of the tables should have between 5 – 10 records, with an average of ~6 records per table. The data should demonstrate relationships properly, and make sense within the scope of the scope of the business problem. You will submit all SQL used, as text. No screenshots.
4. Views – you are required to create 4 views that serve a valid purpose within the business problem. A short, 1 sentence explanation as to the purpose should accompany all views. You will submit all SQL used, as text. No screenshots.
5. Stored procedures and functions – you are required to create at least 1 of each, with a total of 4 objects[3]. Each object should have a short, 1 sentence explanation as to its purpose (which should be valid within the scope of the business problem). You will submit all SQL used, as text.

Important Considerations:

Getting to a working final product is only a part of the battle. Doing so correctly is crucial. About a third of your grade will depend on your code following the SQL standards that we talked about at the beginning of the course. Please make sure you adhere to these.

Tables – you are allowed to use a tool (e.g. Vertabello) to generate the scripts. You are however expected to clean the scripts up to follow our coding standards! Please make sure that the generated SQL contains everything required (identities, PK, FK, nullability and any other constraints applicable), *and give credit since you didn't write most of this SQL*[4].

Data load – I understand it can be a pain loading test data. Please try to keep the data as realistic as possible[5]. Please do not load data such as "Person 1", "Person 2" or "Item 1" or "Item 2".

Views/SPs/Functions – please put some time into these. If all of your objects are very short, and do barely any work, your grade will reflect this negatively. Accordingly, going above and beyond will also

---

[1] I don't mind if you do not follow all of my feedback. You are expected to prevent any data duplication however

[2] No more than 1/3 of a page please.

[3] Doesn't matter to me if it's 1 SP and 3 functions, 2 of each or 3 SPs and 1 function

[4] *Failing to do so is breaking the academic integrity policy and is considered cheating!!*

[5] I usually pick a book/TV show/movie or other data instances from the real world. Internet can be a lot of help in getting some "fake" data that can be used. *Again, if you use an external resource, give credit!*

result in bonus points[6]. I also expect for all of the objects to serve a valid purpose within the business problem. The views should join multiple tables and only select a few columns that are needed to fulfil their purpose. For functions and SPs, on average, I'd like to see at least a 1/3 of a page of code[7].

Deliverable:

Your deliverable will be a *single PDF or .SQL document* containing the design change statement (1) and all of the SQL you wrote for steps (2-5), along w/ the purpose explanations for all objects (in SQL comments). You will submit this document via blackboard.

If you upload a file that does not follow these specifications[8], I reserve the right to lower your grade as I see fit. I may not grade parts of the project (and you will receive 0s for those), or, I may refuse to grade the entire project, and you will receive a 0 for it.

Grading Outline:

1. *Design Modifications (5 pts)*
    a. a list is present
    b. it is not too long
    c. basic thought was put into it
2. *Your design is correct, and no duplicate data is being stored (10 pts)*
3. *SQL Standards followed everywhere (30 pts)*
    a. following naming standards
    b. correct capitalization
    c. correct use of white space (new lines, tabs and so on)
4. *Table creation (15 pts)*
    a. appropriate use of data types and their specification[9]
    b. appropriate use of, and declaration of, identities
    c. appropriate use of constraints (PK, FK, nullable, check, default)
5. *Data load (10 pts)*
    a. following datatypes (text vs numbers)
    b. appropriate number of records loaded
    c. data reflects real world
6. *Views (15 pts), SPs/Functions (15 pts)*
    a. correct number of objects
    b. purpose is stated & accomplished
    c. appropriate complexity

[6] It is up to my discretion in what I consider above and beyond, and how many points this will be worth.
[7] Your goal is to show me what you have learned in class, so please put some effort into these.
[8] Uploading multiple documents or jpegs or using other file formats makes it difficult for me to efficiently grade your project; if you complicate my life enough, I will take points off or give you a 0.
[9] e.g. text allows for appropriate length of data, decimals are declared appropriately and will support correct precision

Design Modifications-

1. *Changed one to many relationship between Classroom table and AVEquipment table to many to many relationship by adding a linking table between Classroom table and AVEquipment table.*
2. *Changed name of ClassDays Table(linking table) to CourseClassDays as two tables within same database cannot have same name. In project 1 ,I had kept name of linking table between ClassDays and ScheduledCourses as ClassDays by mistake.*

Table Creation-

```
CREATE TABLE Person (
	PersonID		INT		PRIMARY KEY	IDENTITY(1,1),
	FirstName		VARCHAR(50)	NOT NULL,
	MiddleName		VARCHAR(50),
	LastName		VARCHAR(50)	NOT NULL,
	DateOfBirth		Date		NOT NULL,
	SSN			CHAR(11)	CHECK(SSN LIKE REPLICATE([0-9],3)+'-
'+REPLICATE([0-9],3)+'-'+ REPLICATE([0-9],3)),
	NTID			VARCHAR(25)	NOT NULL	UNIQUE
);
```

```
CREATE UNIQUE NONCLUSTERED INDEX SSNUnique	--Unique constraint allowing multiple
null values in SSN column
ON Person(SSN)
WHERE SSN IS NOT NULL;
```

```
CREATE TABLE UserRole (
	RoleID		INT			PRIMARY KEY	IDENTITY(1,1),
	Text		VARCHAR(50)		NOT NULL	UNIQUE
);
```

```
CREATE TABLE UserRoles (
	RoleID		INT		REFERENCES UserRole(RoleID),
	PersonID	INT		REFERENCES Person(PersonID),
	PRIMARY KEY(RoleID, PersonID)
);
```

```
CREATE TABLE AddressType (
	AddressTypeID	INT		PRIMARY KEY	IDENTITY(1,1),
	RoleID		INT		NOT NULL	REFERENCES UserRole(RoleID),
	Text		VARCHAR(50)	NOT NULL	UNIQUE
);
```

```
CREATE TABLE Address (
	AddressID	INT		PRIMARY KEY	IDENTITY(1,1),
	PersonID	INT		NOT NULL	REFERENCES Person(PersonID),
	Street1		VARCHAR(150)	NOT NULL,
	Street2		VARCHAR(100),
	City		VARCHAR(50)	NOT NULL	CHECK(City NOT LIKE '%[0-
9!@#$%a^&*()-_+=.,;:'"`~]%'),
	State		VARCHAR(50)	NOT NULL	CHECK(State NOT LIKE '%[0-
9!@#$%a^&*()-_+=.,;:'"`~]%'),
	ZipCode		VARCHAR(10)	NOT NULL	CHECK(ZipCode NOT LIKE
'%[!@#$%a^&*()-_+=.,;:'"`~]%' AND LEN(ZipCode) BETWEEN 3 AND 10)  DEFAULT '000',
```

```
	Country		VARCHAR(50)	NOT NULL	CHECK(Country NOT LIKE '%[0-9!@#$%a^&*()-_+=.,;:'"`~]%'),
	AddressTypeID	INT		NOT NULL	REFERENCES AddressType(AddressTypeID)
);

CREATE TABLE StudentStatus (
	StudentStatusID	INT		PRIMARY KEY	IDENTITY(1,1),
	Text		VARCHAR(50)	NOT NULL	UNIQUE
);

CREATE TABLE StudentAccount (
	StudentID	INT		PRIMARY KEY	REFERENCES Person(PersonID),
	StudentStatus	INT		NOT NULL	REFERENCES StudentStatus(StudentStatusID),
	Password	VARCHAR(20)	NOT NULL	CHECK(LEN(Password) BETWEEN 8 AND 20 AND Password LIKE '%[0-9]% AND Password LIKE '%[A-Z]%' AND Password LIKE'%[0-9!@#$%a^&*()-_+=.,;:'"`~]%')
);

CREATE TABLE College (
	CollegeID	INT		PRIMARY KEY	IDENTITY(1,1),
	Text		VARCHAR(50)	NOT NULL	UNIQUE
);

CREATE TABLE Program (
	ProgramID	INT		PRIMARY KEY	IDENTITY(1,1),
	Text		VARCHAR(50)	NOT NULL	UNIQUE
);

CREATE TABLE StudentSpecialization (
	StudentID	INT		REERENCES StudentAccount(StudentID),
	ProgramID	INT		REFERENCES Program(ProgramID),
	CollegeID	INT		REFERENCES College(CollegeID),
	IsMajor		BIT		NOT NULL		DEFAULT 1,
);

(DAY(EndDate)>=DAY(StartDate) AND MONTH(EndDate)=MONTH(StartDate) AND YEAR(StartDate)=YEAR(EndDate))
		OR (MONTH(EndDate)>MONTH(StartDate) AND YEAR(EndDate)=YEAR(StartDate))
		OR (YEAR(StartDate)<YEAR(EndDate))

CHECK(StartDate<EndDate),
	CHECK((YEAR(EndDate)=YEAR(StartDate) AND (MONTH(EndDate)-MONTH(StartDate)+1)<6)
		OR (((YEAR(EndDate)%YEAR(StartDate)) <> 0) AND (((12-MONTH(StartDate)+1) + ((12*(YEAR(EndDate)%YEAR(StartDate))) -(12-MONTH(StartDate)) <6))))


-- dbo.AgeCalculator returns age of person
CREATE FUNCTION dbo.AgeCalculator(@dateOfBirth AS DATE)
	RETURNS INT
	BEGIN
		DECLARE @result INT
```

```
            SET @result = CAST(DATEDIFF(YEAR,CAST(GETDATE() AS DATE),@dateOfBirth) AS
INT)
            RETURN @result
      END;
```