FIT 100
Assignment 4: SQL

Introduction:
Structured Query Language, or SQL, is the industry standard language for communicating with relational databases. SQL follows a certain pattern and if you know that pattern, you can easily "speak" beginning SQL. The main use of the language in this class will be to view data from various tables and to add or update data.

There are a limited number of clauses, or commands, used by SQL to access and manipulate data stored in a database. The three main clauses are:

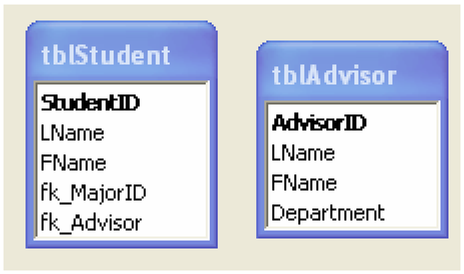
SELECT	Identifies which columns of data in a table are to be displayed
FROM	Identifies which tables hold the data that is needed. Also the location of joins between tables (we'll see that in a second)
WHERE	Non-join criteria that will limit the number of rows to be returned

The outcome of this series of statements will be a table, even if there are no rows of data in it.

The way the tables are associated can be noted in the **FROM** clause using the **INNER JOIN** command. Inner Join is what we mean by the Join term used in the book. It usually indicates a Natural Join. There are other types of joins which is why the term Inner is used to differentiate.

The **WHERE** clause is used to indicate other constraints needed to further limits the rows to be returned.

For example, if an advisor has many students assigned to them, but each student can only have only one advisor, then the association of these two things is one advisor to potentially many students. If Student and Advisor data is to be stored properly in a relational database and the association of a particular advisor with a particular student is also stored, then the table structures would be similar to this one:



AdvisorID is the primary key in **tblAdvisor** and **StudentID** is the primary key in **tblStudent**.

To show that an Advisor is associated with many students, another field is added to tblStudent. This field will hold the key values for the primary key in tblAdvisor and will be considered a foreign key.

RULE: To store a one-to-many relationship between tables, the primary key of the one migrates to become a foreign key in the many. In other words, add another attribute in the table on the many side of the relationship.

Make sure that attribute will allow the same type of data (number, text, etc.) as the primary key attribute of the table on the one side of the relationship. Store the primary key value of the related row in the foreign key attribute.

Then, to display all Student last names and their Advisor last Names, join the tables on the common key attribute and display all rows where the value of AdvisorID in tblAdvisor matches the value of fk_AdvisorID in tblStudent.

```
SELECT      tblStudent.LName, tblAdvisor.LName
FROM        tblStudent INNER JOIN tblAdvisor ON
            tblStudent.fk_AdvisorID =  tblAdvisor.AdvisorID;
```

To *further* limit the rows returned to only those students with the advisor last name of “Smith”, use the Where clause:

```
SELECT      tblStudent.LName, tblAdvisor.LName
FROM        tblStudent INNER JOIN tblAdvisor ON
            tblStudent.AdvisorID =  tblAdvisor.AdvisorID;
WHERE       tblAdvisor.LName = “Smith”;
```

The important points to remember when creating a query are:

- Always include any tables that hold the data you want to see

- Always include any tables that are in the path to the data you want to see.

- When using a join operation, always indicate where the tables are related: Match the keys.

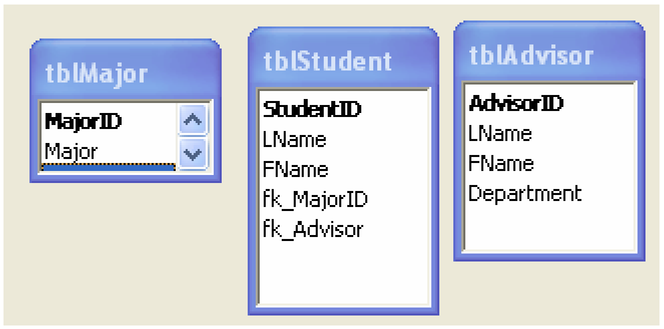
- When generating a query on more than one table, always use table name qualifiers to distinctly identify the correct table.

  Example of table name qualification:

  tblAdvisor.LName and **tblStudent.LName** refer to 2 distinct attributes.

  If you simply use **LName** in a query on those 2 tables, the database query engine will not know which attribute you mean.

Writing Out SQL
Using what you have learned about Structured Query Language, write out the proper SQL syntax in the boxes below to return the requested data. ***You will need to reference commands that were discussed in class***. Use these tables and their attributes as your data sources. You can make the assumption that fk_MajorID and fk_AdvisorID in tblStudent are acting as foreign keys for tblMajor and tblAdvisor.



Example: Show all advisor information from the Advisor table.

```
SELECT      AdvisorID, LName, FName, Department
FROM        tblAdvisor;
```

OR, use the * as a wild card to show all columns:

```
SELECT      *
FROM        tblAdvisor;
```

A) Show all students from the Student table.

B) Show all students in alphabetical order by last name.

C) Show only the Advisor Names from the Informatics Department

D) Show all Advisors and the Students they advise. (Requires a Join)

E) Show the last name and first name of CSE majors. (Requires a Join)

F) Choosing one query from B, C, D, or E do the following:
Explain in English (on back of paper) what each of the SQL commands is doing to retrieve the data you requested. Use the SQL terminology in your answer.

Example for query A:
Select the columns StudentID, LName, FName from the Student table and display them in a virtual table. Bring back all rows, there are no restrictions.