**Name: Bryan Bibb**

Instructions:

Answer the following questions in this document. Do NOT change the question order, numbering, or wording. Submit your completed Journal Entry Assignment via Blackboard. Your Journal Entries are part of the Quiz and Assignments grading category in CPT187 which counts as 15% of your final grade. Refer to the CPT187 Course Syllabus in Blackboard for more information about how your grade in this class is calculated.

IMPORTANT: Please provide code examples whenever possible in your responses to these questions.  
  
Please enter your responses in BLUE so your instructor can see them better. Thanks.

1. (10) In your own words, explain what a relational database is and include the following terms correctly defined in your explanation:

A relational database is an improvement on a system of saving data in flat tables, and began with the work of E.F Codd in 1970. The "structured query language" has improved and diversified over the years but different SQL implementations share a basic data model, that of tables linked together so that the database consists of multiple, overlapping tables that refer to each other. The value of this is that data does not have to be duplicated in the system if it relates to more than one area of information. For example, a customer database will have a customer ID column, which can be linked to the Customer ID column in an Invoices table. Therefore, the Invoices table does not need to include all of the Customer’s personal information: a query can return a search that combines information from both tables as part of the same results set.

A relational database has several fundamental components. First, there are tables that each capture some aspect of the domain, usually modelled on real-world elements. Each of those tables will comprise rows and columns. Each row is a record, i.e., one item in the database, while each column records a different piece of information about the records. Where the row and column intersect is a field, essentially like a cell in a spreadsheet application.

The tool that connects multiple tables is the use of keys. Each table should have one column (usually one; it might be more) defined as the primary key. The primary key column in a table is a unique reference for each record; it cannot repeat. An example would be CustomerID in a Customers table. Another table may then have a column that uses that same set of information defined as the foreign key, such as an Invoices table that has a column where the CustomerID of each invoice is recorded. The CustomerID in this column might repeat a number of times, i.e., for multiple purchases by the same customer. By linking these columns as primary and foreign keys, the database user can perform a search that uses information in both tables.

A database’s tables and their relationships (along with many other ‘objects’ that can live in the database such as views, triggers, and stored procedures) are maintained in a DBMS, databse management system. The DBMS consists of a backend server that uses a certain dialect of SQL, such as MSSQL’s Transact-SQL, which is accessed with an interface application that searches, views, and adds/updates information in the system.

* + database management system (DBMS)
  + column
  + row
  + table
  + record
  + field
  + cell
  + primary key
  + foreign key

1. (3) Define and explain the difference in one-to-many relationship, many-to-many relationship, and one-to-one relationship.

The most common type of relationship is one-to-many, illustrated by the CustomerID example above. Each piece of data can occur once in the primary key column, but many times in the foreign key column.

A one-to-one relationship would be where the data in the primary key column links specifically to one bit of data in the foreign key column. This relationship could therefore be combined into one table. It would be like if the CustomerAddress table had a CustomerID column with customer addresses, while CustomerContact table had an email and phone number for each CustomerID. There may be a reason to have the tables separated, but it would probably be better to combine them and use defined views if the data needs to be limited in its access for some reason.

A many-to-many relationship is basically like a JOIN: two tables have a column with data that can repeat (say, CustomerIDs in an Invoices table and in a CustomerOrders table). For this work, each of those tables must have a one-to-many relationship with an intermediate third table where the Primary key appears only once. The Invoices table and CustomerOrders table would use the Primary key of CustomerID in the Customer table to sync up.

1. (15) Define the following terms from Chapter 17 in our Murach textbook:
   * data type

The data type determines what kinds of data can be used in a column, similar to the datatype of an object. Common types are TEXT, INTEGER, REAL, and BLOB.

* + null value

Null value means the field does not have any data in it. When a table is defined, columns can be labeled as whether they accept NULL, which means whether they can be empty at the time a record is created. For example, a customer’s last name would not allow NULL, but their ‘suffix’ column would because not every name needs it.

* + default value

Default value refers to the data that is entered into the field at the time that the record is created if no data is provided. For example, a default InvoiceTotal value of $0 might be put into the table so that a record can be created with no Total in a column that does not allow NULL values.

* + orphaned key

An orphaned key is a foreign key that has had its primary key in another table deleted. For example, if a user record were deleted from the Customer table, any reference to that CustomerID in Invoices would be orphaned. When tables are being created, “CHECK” statements can enforce “key integrity,” which would prevent you from deleting that Customer record if any other table depended on it.

* + DML

Data Manipulation Language refers to statements that search the database, display results, and add/update/delete records as needed. These include SELECT, INSERT, UPDATE, and DELTE.

* + DDL

Data Definition Language refers to statements that set up the underying data structure of the database, including commands to CREATE DATABASE, DROP DATABASE, and DROP TABLE. Generally these will only be used by database designers and administrators, while database users can perform DML statements.

* + SQL

Structured Query Language refers to the standard language used by SQL implementations. There are differences in the SQL dialect used by Microsoft, Oracle, Postgres, MySQL, etc., though the basics are the same.

* + Query

Despite its name, a query is not only a search statement. It can refer to a SELECT statement that searches the database for a particular subset of information, but any SQL commands can be run (both DML and DDL) in a ‘query’ pane.

* + result set

A result set refers to the data set returned by a query.

* + result table

A result table is the visual representation of a result set. If a SELECT command is successful, a table of information will show the requested data in the console.

* + join

A join is SELECT statement that links two or more tables together through their keys for the purpose of searching that larger superset of data at once.

* + inner join

There are multiple ways of performing a JOIN but an INNER JOIN is the most common, in which records are included in the results only if their related columns match. Records where that column is empty are not included. The INNER keyword is not required, and an inner join is performed with the default JOIN statement.

* + alias (as used in a query)

An alias is an alternative name given to a column in a SELECT statement. You might use an alias if two columns in a JOIN condition have the same name, or you want to provide a more user-friendly name than the one in the database. You also use aliases when a column is the result of some combination of columns or a calculation.

* + connection object

The connect() method of the sqlite3 modules creates a connection object which you use to access the database. It should be closed with the close() method when the connection is finished.

conn = sqlite3.connect(“movies.sqlite”)

…

if conn:

conn.close()

* + cursor object

A cursor object is created by the cursor() method. In SQL a cursor is a temporary object in the database that is used to read and manipulate data, or basically a temporary workspace. It points to one row at a time but it can iterate through rows in a query result. In sqlite3, the cursor receives the data from the query, and the class also provides methods to operate on that data, including execute(), fetchone(), and fetchall().

1. (3) Using Python, write a block of code that is a good example of an INSERT statement query.

This adds Oppenheimer to the Movies database as a new record:

name = "Oppenheimer"

year = "2023"

minutes = 180

categoryID = 2

with closing(conn.cursor()) as c:

sql = '''INSERT INTO movie (name, year, minutes, categoryID)

VALUES (?, ?, ?, ?)'''

c.execute(sql, (name, year, minutes, categoryID))

conn.commit()

1. (3) Using Python, write a block of code that is a good example of an UPDATE statement query.

This changes the categoryID of Oppenheimer to category 3:

categoryID = 3

name = "Oppenheimer"

with closing(conn.cursor()) as c:

sql = '''UPDATE Movie

SET categoryID = ?

WHERE name = ?'''

c.execute(sql, (categoryID, name))

conn.commit()

1. (3) Using Python, write a block of code that is a good example of a DELETE statement query.

This deletes the Oppenheimer record from the database

name = "Oppenheimer"

with closing(conn.cursor()) as c:

sql = '''DELETE FROM Movie

WHERE name = ?'''

c.execute(sql, (name,))

conn.commit()

1. (1) Why does the Murach textbook like using SQLite to manage Python databases?

SQLite is open-source, easy to setup, and benefits from built-in support in Python. Therefore, it is easy to use in projects by just importing the sqlite3 module, and no backend server is required.

(38 possible points)