DATA MANAGEMENT - APPLICATIONS

Competencies:
4018.1.1: Conceptual Models to Physical Schemas - *The graduate creates conceptual data models and translates them into physical schemas.*
4018.1.2: Create Databases - *The graduate creates databases utilizing SQL Data Definition Language (DDL) in MySQL environment.*
4018.1.3: Create/Modify Tables and Views - *The graduate creates and modifies tables and views employing SQL Data Definition Language (DDL) in MySQL environment.*
4018.1.4: Create Primary Keys/Foreign Keys and Indexes - *The graduate creates and modifies primary keys (PKs) and foreign keys (FKs) and indexes with SQL Data Definition Language (DDL) in MySQL environment.*
4018.1.5: Populate Tables - *The graduate populates tables with insert, update, and delete using DML in the MySQL environment.*
4018.1.6: Create Simple and Complex Queries - *The graduate creates simple Select-From-Where (SFW) and complex 3+ table join queries with Data Manipulation Language (DML) in MySQL environment.*

Introduction:

For this assessment, you will be creating models, databases, tables, and query reports for a smartphone application.

To complete this assessment, you will be using SQL Fiddle, an online SQL tool, to test run the MySQL code that you will develop for parts C through G. After running the code, you will take a screenshot of your results, which should be copied into the document that you will submit to TaskStream for this assessment. Instructions for how to use this tool for each part of the assessment is included in the attached document "SQL Fiddle Instructions." Please note that for each part of the assessment, there are explicit instructions on what SQL code you will need to copy and paste into SQL Fiddle panels to run your test.

The work you complete for each part of the assessment (i.e., the design models/diagrams, tables, written explanations, SQL script code, and screenshot results from running your SQL scripts in SQL Fiddle) should be saved as a single *.pdf (Portable Document Format) file that you will submit to TaskStream.

Scenario:

You are the database designer and developer for a donut shop that wants to create a smartphone application where customers can order donuts. First, you will design a normalized entity-relationship (E-R) logical database model to store data related to the customer, donuts, and donut order. Next, you will create three tables with primary and foreign keys that are derived from your E-R model. Once the tables have been built, then you will create views and indexes to protect and fine-tune query performance. You will populate each of the tables with sample data. Finally, you will create both a simple "select-from-where" (sfw) query and a complex join query to produce meaningful reports on individual donut orders and summaries to determine which donuts sell the best.

Requirements:

A. Construct a normalized model to represent the donut shop smartphone application database that supports the scenario above by doing the following:
   1. Using the unnormalized order form ("Sales Order Form" attached below) for the donut ordering database as a reference, produce the final logical schema for this database by doing the following:

Note: You can design the tables using any method of your choice (e.g., using a drawing tool using tables).

a. Design a table that is in **1st normal form** and fulfills the following requirements:
   - The table should have a primary key that uniquely identifies the records.
   - The values in each of the columns should be atomic.
   - There should be no repeating groups.

   i. Write a **one** paragraph explanation of how you designed the table.

b. Design *at least* **two** tables that are in **2nd normal form** and fulfill the following requirements:
   - The tables should meet all of the requirements for the 1st normal form.
   - Redundant data appearing in multiple rows should be eliminated with the creation of the new tables.
   - Each of the tables should be related using foreign keys.

   i. Write a **one** paragraph explanation of how you designed the table

c. Design *at least* **three** normalized tables that are in **3rd normal** form and fulfill the following requirements:
   - The tables should meet all of the requirements for the 1st and 2nd normal forms.
   - Fields that do not depend on the primary key or any other field should be eliminated.
   - There should be at least three resulting tables, one for customer information, one for donut information, and one for order information.

   i. Write a **one** paragraph explanation of how you designed the table

B. Using the tables you designed in **3rd normal** form from part A1c, create an entity-relationship (E-R) diagram that fulfills the following requirements:

Note: You can use any drawing tool of your choice to create the diagram.

- Entities should be drawn that represent *each* of the tables from the 3rd normalized form.
- All appropriate fields (i.e., attributes) should be entered into each of the entities.
- Primary keys (PK) and foreign keys (FK) should be appropriately designated.
- Data types for *each* attribute (Numeric, Fixed, Char, Varchar, or Timestamp) should be appropriately designated.
- Relationships drawn between the entities should be labeled with a relationship name.

1. Provide a written explanation for the following:
   a. Explain why you selected the entities represented in your diagram.
   b. Explain how you determined the relationships between these entities.
   c. Explain what the cardinality is for *each* of the relationships.

C. Develop the SQL code for *each* of the tables you designed in part A and refined in part B by doing the following:

1. Provide the SQL code you wrote for *each* table.

Note: Make sure all of the primary and foreign keys and data types are assigned to each of the fields in the create table SQL code.

a. Demonstrate that you have used SQL Fiddle to test your code from part C1 by providing a screenshot of your results.

D. Using the table that contains the customer information fields, develop SQL code to create a View that shows all of the customer information with the First Name and Last Name concatenated (CONCAT()) to show full name as one field by doing the following:

1. Provide the SQL code you wrote to create the View for the customer information.
   a. Demonstrate that you have used SQL Fiddle to test your code from part D1 by providing a screenshot of your results.

E. Using the table that contains the donut information fields, develop SQL code to create an Index for the donut name field by doing the following:
   1. Provide the SQL code you wrote to create the Index for the donut information.
      a. Demonstrate that you have used SQL Fiddle to test your code from part E1 by providing a screenshot of your results.

F. Using all of the tables you developed in part C, develop SQL code to populate *each* of the tables by doing the following:
   1. Provide the SQL code that inserts data into all of the tables.

   Note: Make sure that data is inserted first into the table(s) with primary keys before inserting data into the other tables.

      a. Demonstrate that you have used SQL Fiddle to test your code from part F1 by providing a screenshot of your results.

G. Using the tables you populated in part F, develop SQL code to display the values in a requested table or tables by doing the following:
   1. Provide the SQL code for the simple (sfw) queries to display all of the data in each of the tables you have created and populated.
      a. Demonstrate that you have used SQL Fiddle to test your code from part G1 by providing a screenshot of your results.
   2. Provide the SQL code for a complex join query to display all of the information contained in the attached "Sales Order Form."
      a. Demonstrate that you have used SQL Fiddle to test your code from part G2 by providing a screenshot of your results.

H. Submit parts A–G as a *.pdf (Portable Document Format) to TaskStream.

*Note: You can use any word-processing or other program of your choice to compile each part of the assessment. Please clearly label each part. Please save your document as a *.pdf (Portable Document Format) file before submitting to TaskStream.*

Note: For definitions of terms commonly used in the rubric, see the Rubric Terms web link included in the Evaluation Procedures section.

File Attachments:

1. **"Sales Order Form"**
2. **"SQL Fiddle Instructions"**

Web Links:

1. **SQL Fiddle**