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 MySQL to test and run a database application that you will develop for parts C through G.

After running the code, take a screenshot of your results and paste the screenshot into the document that you will submit to TaskStream for this assessment.

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 a SQL tool) should be saved as a single *.pdf (Portable Document Format) file that you will submit to TaskStream.

Note: If you do not have access to a database tool, you may use SQL Fiddle (an online SQL tool) to complete this assessment. The tool can be accessed at the following link: http://sqlfiddle.com. Instructions for how to use SQL Fiddle for each part of the assessment are 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.

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 four 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 attached "Sales Order Form" (unnormalized order form) 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 for the following prompts using any method of your choice (e.g., using a drawing tool using tables).

a. Design **one** table that is in first 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.
   - Do not include the calculated attributes/fields (Line Total, Subtotal, Sales Tax and Total) in your normalization diagrams.
     i. Explain how you designed the table *(suggested length of 1 paragraph).*

b. Design **three** tables that are in second normal form and fulfill the following requirements:
   - The tables should meet *all* of the requirements for the first normal form.
   - All functional dependencies should be removed from the tables.
   - *Each* of the tables should have *all* primary and/or foreign keys designated.
   - Do not include the calculated attributes/fields (Line Total, Subtotal, Sales Tax and Total) in your normalization diagrams.
     i. Explain how you designed the tables (*suggested length of 1 paragraph*).

c. Design **four** normalized tables that are in third normal form and fulfill the following requirements:
   - The tables should meet *all* of the requirements for the first and second normal forms.
   - *All* transitive dependencies should be removed from the tables.
   - *Each* of the tables should have all primary and/or foreign keys designated.
   - Do not include the calculated attributes/fields (Line Total, Subtotal, Sales Tax and Total) in your normalization diagrams.
   - There should be **four** resulting tables: one for customer information, one for donut information, one for order information and one for order line item information.
     i. Explain how you designed the tables *(suggested length of 1 paragraph).*

B. Create an entity-relationship (E-R) diagram, using the tables you designed in third normal form from part A1c, that fulfills the following requirements:

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

   1. Draw entities to represent *each* of the tables from the third normalized form.
   2. Enter *all* appropriate fields (i.e., attributes) into *each* of the entities.
      a. Designate primary keys (PK) and foreign keys (FK).
      b. Designate data types for *each* attribute (i.e., Numeric, Fixed, Char, Varchar, or Timestamp).
   3. Label relationships drawn between the entities with a relationship name.

      Note: The cardinality of each relationship needs to be identifiable.

   4. 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 the types of relationships (i.e., cardinality) used in your diagram.

C. Develop the SQL code to create *each* of the third normal form 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.
      a. Demonstrate that you have tested your code from part C1 by providing a screenshot of your results.

D. 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 using the table that contains the customer information fields by doing the following:
   1. Provide the SQL code you wrote to create the “View” for customer information.

a. Demonstrate that you have tested your code from part D1 by providing a screenshot of your results.

E. Develop SQL code to create an “Index” for the donut name field using the table that contains the donut information fields by doing the following:
1. Provide the SQL code you wrote to create the “Index” for the donut information.
   a. Demonstrate that you have tested your code from part E1 by providing a screenshot of your results.

F. Develop SQL code to populate *all* of the tables developed in part C 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 tested your code from part F1 by providing a screenshot of your results.

G. Develop SQL code to display the values in a requested table or tables using the tables populated in part F 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 tested 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” with the exception of the calculated attributes (Line Total, Subtotal, Sales Tax and Total).
   a. Demonstrate that you have tested 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:

Sales Order Form

SQL Fiddle Instructions

Web Links:

1. **SQL Fiddle**
2. **VHT Task 1 Rubric**