COURSEWORK ASSIGNMENT

UNIVERSITY OF EAST ANGLIA
School of Computing Sciences

Module: CMP-4010B
ASSIGNMENT TITLE: SQL and Python Programming Exercise

DATE SET	**:**	**Week 7**	
DATE & TIME OF SUBMISSION	**:**	**Thursday, 23rd April week 12** **Demonstration Week 12 (from Monday)**	
RETURN DATE	**:**	**Assessment Period Week 4**	
ASSIGNMENT VALUE	**:**	**40 %**	
SET BY	**:**	**Dr Beatriz de la Iglesia**	**SIGNED:**
CHECKED BY	**:**	**Dr Graeme Richards**	**SIGNED:**

Aim:
To implement a database application in Python by first completing the database table definitions and writing interactive SQL statements.

Learning outcomes:
Experience in the following:
- problem solving techniques using Python, SQL and postgreSQL;
- interpreting user requirement and defining solutions;
- creating table definitions using SQL;
- manipulating table data using SQL;
- SQL programming in postgreSQL;
- managing a database from a Python application using Psycopg2
- demonstration and presentation of technical systems;
- managing time based on workload, deadlines and distribution of effort.

Assessment criteria
Good use of SQL data definition language to complete the table definitions;
Good use of SQL data manipulation language to write interactive queries;
Good use of Python to implement a prototype database client application;
Ability to correctly and accurately interpret project specification;
Correct functionality and output as required by each requirement;
Neatly presented work with correct program output during demonstration.

Description of assignment:
See Attached.

Required:
See Attached

Handing in procedure:

You will be expected to demonstrate your work in week 12. The SQL and program demonstrated must match the version handed in! All deliverables will be submitted electronically (instructions on how to submit electronically will be issued in due course).

Plagiarism:
Plagiarism is the copying or close paraphrasing of published or unpublished work, including the work of another student without the use of quotation marks and due acknowledgement. Plagiarism is regarded a serious offence by the University and all cases will be reported to the Board of Examiners. Work that contains even small fragments of plagiarised material will be penalised.

UNIVERSITY OF EAST ANGLIA
School of Computing Sciences

CMP-4010B Database Systems

Coursework Assignment 2 – SQL and Python/Flask Programming

Set: Week 7
Hand in: Thursday 23rd April, Week 12 (Demonstrations to be held anytime in Week 12)
Returned by: Assessment period week 4
Value: 40% of unit marks

Introduction

You have been given the opportunity to implement a database system for a software company. The company has a number of products and is in the process of setting up web and phone support for its customers. The database will contain a ticket system to track customer support issues. The ticket system works by customers logging support questions which are stored in the database. Staff members respond to these questions and log an update to the ticket. When the question has been suitabley answered and no further response has been received from the customer, the ticket is closed. Your role is to prototype and test some of the functionality required for the backend system.

The first stage in this process is to analyse the requirements and write SQL statements to perform these tasks. These statements can be tested using an interactive SQL interface to ensure correct functionality before writing the programmatic interface. Once the correct functionality has been achieved, you are required to develop a Python application to execute SQL statements. The Python application should use Psycopg2 to execute the commands developed as a result of using the interactive SQL interface. Your prototype application will be used to demonstrate the processes and database functionality to the company before a full desktop and web user interfaces are developed; therefore you only need to implement the features required for the user interaction described in the tests below.

A description of the tables and required functionality has been provided. Naturally it is grossly simplified compared with a real system. A detailed specification of the task to be undertaken and the deliverables to be produced for assessment are given below.

System Functionality

The database comprises the following tables:

Staff (StaffID, Name)
Product (ProductID, Name)
Customer (CustomerID, Name, Email)
Ticket (TicketID, Problem, Status, Priority, LoggedTime, CustomerID* , ProductID*)
TicketUpdate (TicketUpdateID, Message, UpdateTime, TicketID* , StaffID*)

where *Status* can be either '*open' or 'closed'* and *Priority* is represented by the numbers 1-3 (with 1 being the highest priority).

A number of assumptions have been made for you, these are:

- Only the tasks below need to be provided to the user. All other tables and data can be managed using interactive SQL, e.g. managing staff and products.
- It is not necessary to auto-generate ID numbers for the tables. IDs will be supplied when data is provided for testing and assessment.
- The user interface can be console based or web based but most of the marks will be awarded based on the ability to complete the tasks and present the results clearly, not on how advanced the user interface is.
- You will need to provide validation and integrity checks for the tasks required by the prototype.

- A null value in the Staff ID field in the TicketUpdate table represents the update has come from the customer and not a member of staff (the customer ID can be acquired through the TicketUpdate-Ticket-Customer relationship).
- You do not need to worry about product registration, i.e. the relationship between Customer and Product.

The actions of interest to us in this work are:

.

1. Given a CustomerID, name and email, create a new Customer record.
2. Create a new support ticket for a customer with respect to a given product. The application should display details of the ticket as confirmation.
3. Add an update to the support ticket from a given staff member.
4. List all outstanding (i.e. open) support tickets along with the time of the last update (if applicable).
5. Given a ticketID, set the status of a support ticket to closed.
6. Given a ticketID, list the original problem as the question along with all updates in chronological order. For each line, include the time of the message and the name of the person who authored the message.
7. Produce a report showing the status of each closed support ticket. For each ticket you should include the number of updates, the time that elapsed between the question being logged and the first update, and the time between the question being logged and the final update. *You may assume that all closed tickets have at least one update from a member of staff.*
8. Close all support tickets which have not been updated for at least 24 hours. This will be records that have received at least one update from a staff member and no further updates from the customer (or staff member) for at least 24 hours.
9. Given a Customer ID, permanantly remove the customer's details. The removal of a customer should not be possible while the customer is associated with any tickets.

Assignment

Part 1. Create a test database (25% of marks)

A copy of the create table statements for the database definition is available in a text file CW2Tables.txt in Blackboard. Prepare additional SQL clauses and/or statements to complete the definition of the database by specifying primary keys, domain constraints, entity and referential integrity constraints, etc. Note that you should NOT modify the name and type of the attributes (i.e. the information you have been given). Save all your Data Definition Language (DDL) statements in a text file.

Load a reasonable volume of test data into the tables for use in your testing. The test data should be sufficient to test all of the queries with their expected output and should provide a suitable environment in which to test normal operation as well as abnormal conditions.

- Document this stage with a copy of your complete DDL statements in SQL (including any table definitions, views, triggers, comments, etc.) **You should bring a printed copy to the demonstration.**
- Produce also a copy of the test data you loaded for testing.

Part 2. Test the functionality (50% marks aprox.)

For each of the tasks described above, prepare in text files interactive SQL statements (and comments). Test these statements using the PGAdmin III interface. You should ensure you test your queries thoroughly, e.g. test entity integrity, referential integrity and other constraints as necessary.

The purpose of this stage is to prototype and test SQL statements for each task for use in your final version of the Python application. However, your .sql files need to ready to be loaded during the demo so if you fail to demonstrate your Python program working you can at least demonstrate your prototype SQL statements through the interactive interface.

- Document this stage with a copy of the SQL statements. **Your SQL statements need to be accessible (through copy/paste) as text files through the demonstration.**

- Evidence of testing of each SQL statement (e.g. copy of the output of running the query).

Part 3. Develop a prototype version of the client (25% marks aprox.)

Write a Python/Flask/Psycopg2 application. The application should comprise a home web page with a number of forms each of which handles one of the tasks above. Each form's action should lead to a Python function to execute the relevant SQL query. If the query requires output this should be presented on another web page which should include a link back to the home page. If an error occurs a suitable error message should be displayed on the home page. When a task is complete, a simple one line message should be displayed. Python must not be used for any data processing other than submitting SQL statements and receiving results.The objective is to demonstrate the good use of Python and SQL for database acces. Whilst your program should be well laid out and easily readable, no extra credit will be given for complex coding and exotic user interfaces are not required!

- The deliverable for this part of the assignment will be a copy of an *annotated* program source listing (your .py file) and the contents of your templates folder. You **do not** need to produce a printed copy of your Python code for the demonstration.

All deliverables from part 1, 2 and 3 should be collated together in a folder which should then be zipped ready for electronic submission by the **coursework deadline**. Instruction for electronic submission of code will be place on Blackboard.

Week 12 Demonstration of Part 3

Please *keep* your tables, test data, interactive SQL statements and program after completing the assignment so that you can give a demonstration to show your Java application working. Demonstrations will be scheduled for week 12 and **all marks will be awarded at the demonstration**.

Demonstrating your work is therefore a *mandatory* part of the assessment procedure and those that do not come for their scheduled demonstration time will receive a mark of zero for the assignment unless they have obtained an extension to the deadline. The demonstration will involve loading a set of *provided* test data and carrying out a standard set of tests. The SQL and program demonstrated must match the version submitted!

Important Notes

- The documents should be neatly formatted to ease reading.
- This is an individual piece of coursework NOT a group project. Collusion/plagiarism checks may be carried out.
- As you will be given a SQL script in week 12 to load test data into your tables, it is vital that you do not change the table names, field names, field types etc. from what is described below.
- The demonstration must take place on a CMP lab machine using a Python desktop application as the client communicating with the CMP PostgreSQL database server. We will not accept other systems, languages, technologies or machines.

Summary of Deliverables

- Electronic submission to Blackboard:
    - Part 1: A copy of your SQL data definition statements, annotated as necessary, together with the test data.
    - Part 2: Your SQL data manipulation statements (annotated with comments if necessary) for each of the requirements plus evidence of testing.
    - Part 3: Your python source code files (.py).
- Attend and participate in a demonstration in week 12.

Appendix

Initial database design

```
CREATE TABLE Staff
(
      ID          INTEGER,
      Name        VARCHAR(40)
)

CREATE TABLE Product
(
      ID          INTEGER,
      Name        VARCHAR(40)
)

CREATE TABLE Customer
(
      ID          INTEGER,
      Name        VARCHAR(40),
      Email       VARCHAR(40)
)

CREATE TABLE Ticket
(
      ID          INTEGER,
      Problem     VARCHAR(1000),
      Status      VARCHAR(20),
      Priority    INTEGER,
      LoggedTime  TIMESTAMP,
      CustomerID  INTEGER,
      ProductID   INTEGER
)

CREATE TABLE TicketUpdate
(
      ID          INTEGER,
      Message     VARCHAR(1000),
      UpdateTime  TIMESTAMP,
      TicketID    INTEGER,
      StaffID     INTEGER
)
```