CPSC 332 Final Project Report

Group Members:

Alberto Rodriguez
Angel Quiroga
Austin Kemper

How to Initialize and Run the Database:

1. Start up XAMPP and its Apache and MySQL servers
2. Run the DocOffice.py program. This program will create the database, its tables, and input data for each.
3. `Views.sql` is a collection of SQL scripts meant for fulfilling the queries assigned to us for this assignment. To run, you may copy and paste each script into the XAMPP query field and execute to show the query results

Explanations of Files:

DocOffice.py:

```
#DocOffice File
import pymysql

#database connection
#connection = pymysql.connect(host="localhost", user="root", passwd="", database="database")
conn = pymysql.connect(host="localhost", user="root", passwd="")

cursor = conn.cursor()
cursor.execute('DROP DATABASE IF EXISTS DocOffice')
#conn.cursor().execute('DROP DATABASE IF EXISTS DocOffice')
cursor.execute('CREATE DATABASE DocOffice')
cursor.execute('USE DocOffice')


# Queries for creating table
createDoctor = """CREATE TABLE Doctor(
  DoctorID              VARCHAR(6)          NOT NULL,
  MedicalDegrees    VARCHAR(50)                 ,
```

```
84        PRIMARY KEY (PersonID))"""
85
86
87
88    # Creates tables in database
89    cursor.execute(createPerson)
90    cursor.execute(createDoctor)
91    cursor.execute(createPatient)
92    cursor.execute(createPrescription)
93    cursor.execute(createTest)
94    cursor.execute(createPatientVisit)
95    cursor.execute(createSpecialty)
96    cursor.execute(createDoctorSpecialty)
97    cursor.execute(createPVisitPrescription)
98
99
100   # Populate tables with dummy data
101   populateDoctor = """INSERT INTO Doctor
102     VALUES
103       ('RO3283', 'Pediactrics', 'RO3283'),
104       ('AR3456', 'Neurology', 'AR3456'),
105       ('WF3421', '', 'WF3421')"""
106
```

```
145       ('MOUTH', 'Teeth')"""
146
147   populatePerson = """INSERT INTO Person
148     VALUES
149       ('RO3283', 'Rob', 'Belkin', '800 State College', 'Fullerton', 'California', '90643', '456
150       ('AR3456', 'Alan', 'Rickman', '410 El Rancho', 'La Habra', 'California', '90631', '626456
151       ('RM1234', 'Robert', 'Morris', '320 Shady Lane', 'Yorba Linda', 'California', '90123', '5
152       ('MR4567', 'Martin', 'Rodriguez', '540 Painter Ave', 'Whittier', 'California', '90893', '
153       ('WF3421', 'Winston', 'Franks','310 West Ave', 'Whittier', 'California', '90324', '435675
154
155   cursor.execute(populatePerson)
156   cursor.execute(populateDoctor)
157   cursor.execute(populatePatient)
158   cursor.execute(populatePrescription)
159   cursor.execute(populateTest)
160   cursor.execute(populatePatientVisit)
161   cursor.execute(populateSpecialty)
162   cursor.execute(populateDoctorSpecialty)
163   cursor.execute(populatePVisitPrescription)
164
165
166
167
```

The DocOffice.py program utilizes the `pymysql` module that allows easy and simple connections to mySQL servers. After connecting to the database using `.connect()`, a cursor is made to aid in executing SQL scripts for the project (see Image 1).

The cursor is set to variable cursor and is used with `cursor.execute()` in order to execute the programmed SQL scripts (see Images 2 & 3).

Views.sql:

```
/* Number 2:  Doctor Rob Belkin is retiring. We need to inform all his patients, a
select a new doctor. For this purpose, Create a VIEW that finds the names and
Phone numbers of all of Rob's patients.
*/
CREATE VIEW Rob_Patient
AS
SELECT p.FirstName, p.LastName, p.PhoneNumber
FROM Person AS p
INNER Join Patient AS pt
ON pt.PersonID = p.PersonID

INNER JOIN PatientVisit pv
ON pv.PatientID = pt.PatientID
WHERE pv.DoctorID IN (SELECT d.DoctorID FROM Doctor AS d
                      INNER JOIN Person AS pr ON pr.PersonID = d.PersonID
                      WHERE pr.FirstName = "Rob" AND pr.LastName = "Belkin")
```

Script #2 joins the Patient and Patient Visit tables through an `INNER JOIN.` It pulls solely records that have Rob Belkin's information and returns the query.

```
/* Number 3: Create a view which has First Names, Last Names of all doctors who gave out
prescription for Panadol.
*/
CREATE VIEW Pandol_Doctor
AS
SELECT p.FirstName, p.LastName, pr.PrescriptionName
From Person AS p
INNER JOIN Doctor AS d
ON p.PersonID = d.DoctorID
INNER JOIN PatientVisit AS pv
ON pv.DoctorID = d.DoctorID
INNER JOIN PVisitPrescription AS pp
ON pv.VisitID = pp.VisitID
INNER JOIN Prescription AS pr
ON pr.PrescriptionID = pp.PrescriptionID
WHERE pr.PrescriptionName = "Pandol";
```

Script #3 Does several `INNER JOINS` between Person, PatientVisit, PVisitPrescription, and Prescription to search for where the PrescriptionName is Pandol.

```
/* Number 4: Create a view which shows the First Name and Last name of all doctors and their
specialty's.
*/

CREATE VIEW Doctor_Specialities
AS
SELECT p.FirstName, p.LastName, spec.SpecialtyName
FROM Person as p
INNER JOIN DoctorSpecialty as docSpecial
ON p.PersonID = docSpecial.DoctorID
INNER JOIN Specialty as spec
ON docSpecial.SpecialtyID = spec.SpecialtyID

/*Number 5: Modify the view created in Q4 to show the First Name and Last name of all
doctors and their specialties ALSO include doctors who DO NOT have any
specialty. */

SELECT p.FirstName, p.LastName, spec.SpecialtyName
FROM Doctor as d
LEFT JOIN Person as p
ON d.PersonID = p.PersonID
LEFT JOIN DoctorSpecialty as docSpecial
ON d.DoctorID = docSpecial.DoctorID
LEFT JOIN Specialty as spec
ON docSpecial.SpecialtyID = spec.SpecialtyID
```

Scripts #4 & #5 work similarly where they use several `JOINS` to combine the Doctor, Person, DoctorSpeciality, and Specialty tables to view all the needed fields. #5 differs in that it utilizes `LEFT JOINS` to make all Doctors in the Doctor table displayed, regardless whether or not they have a speciality

```
/*Number 6: Create trigger on the DoctorSpecialty so that every time a doctor
            specialty is updated or added, a new entry is made in the audit table.
*/
CREATE or REPLACE TRIGGER Audit_change
AFTER
UPDATE[of Audit]
ON DoctorSpecialty
BEGIN
    INSERT INTO Audit VALUES ('Doctor specialty is updated', sysdate);
END;

CREATE TRIGGER update_audit
AFTER UPDATE ON DoctorSpecialty
FOR EACH ROW
BEGIN
    INSERT INTO sp_audit
    DoctorID = OLD.DoctorID,
    SpecialtyID = OLD.SpecialtyID;
END;
```

Script #6 creates a trigger to generate a new entry into the table when DoctorSpeciality gets updated or added to

```
/* Number 7:  Create a script to do the following (Write the script for this)
            a. If first time backup take backup of all the tables
            b. If not the first time remove the previous backup tables and take new
                backups.

*/
BACKUP DATABASE DocOffice
TO DISK = '"C:\Users\'
```

Script #7 creates a backup of the DocOffice database and saves it under C: