**CSC 455: Database Processing for Large-Scale Analytics**

**Assignment 2**

**Due 11:59pm, Wednesday, October 9th.**

**Part 1**

**You are given a following schema (First, Last, Address, Job, LNumber, Interest, Amount) and the following set of functional dependencies:**

**First, Last 🡪 Address**

**LNumber 🡪 Interest, Amount**

**(remember that underlined primary key implies First, Last, Job, LNumber 🡪Address, Interest, Amount)**

**Decompose the schema to make sure it is in Third Normal Form (3NF).**

**Write SQL DDL and SQL INSERT statements to create the 3NF tables and load the data provided in the text file data\_hw2.txt (posted in the documents section on COL). Note that NULL is supposed to represent database NULL – in some cases NULL may be represented by another special symbol to distinguish it from the ‘NULL’ string.**

**ANSWER:**

**1.**

(First, Last, Job, LNumber)

(First, Last, Address)

(LNumber, Interest, Amount)

**2.**

DROP TABLE ACCOUNTS;

DROP TABLE CLIENTS;

DROP TABLE LOANS;

CREATE TABLE Clients

( First Varchar2(10),

Last Varchar2(10),

Address Varchar2(30),

CONSTRAINT Clients\_PK

PRIMARY KEY(First,Last)

);

CREATE TABLE Loans

( Lnumber Number(2) PRIMARY KEY,

Interest Number(5,2),

Amount Number(10)

);

CREATE TABLE Accounts(

First Varchar2(10),

Last Varchar2(10),

Job Varchar(20),

Lnumber Number(5),

CONSTRAINT AccountsFK

FOREIGN KEY (First,Last)

REFERENCES Clients(First,Last),

CONSTRAINT AccountsFK2

FOREIGN KEY (Lnumber)

REFERENCES Loans(Lnumber),

CONSTRAINT Accounts\_PK

PRIMARY KEY(First,Last,Job,Lnumber)

);

INSERT INTO Loans values(5,1.0,200);

INSERT INTO Loans values(1, 5.0, 10000);

INSERT INTO Loans values(2, 11.8, 7000);

INSERT INTO Loans values(7, 10.0, 100);

INSERT INTO Loans values(8, 20.0, 25000);

INSERT INTO Loans values(4, 3.3, 100000);

INSERT INTO Loans values(3, 2.0, 12000);

INSERT INTO Clients values('John','Smith','111 N. Wabash Avenue');

INSERT INTO Clients values('Jane','Doe','243 S. Wabash Avenue');

INSERT INTO Clients values('Mike','Jackson','1 Michigan Avenue');

INSERT INTO Clients values('Mary','Who','NULL');

INSERT INTO Accounts values('John','Smith','plumber',5);

INSERT INTO Accounts values('John','Smith','bouncer',5);

INSERT INTO Accounts values('Jane','Doe','waitress',1);

INSERT INTO Accounts values('Jane','Doe','accountant',2);

INSERT INTO Accounts values('Jane','Doe','waitress',2);

INSERT INTO Accounts values('Jane','Doe','accountant',1);

INSERT INTO Accounts values('Mike','Jackson','accountant',7);

INSERT INTO Accounts values('Mike','Jackson','accountant',8);

INSERT INTO Accounts values('Mike','Jackson','accountant',4);

INSERT INTO Accounts values('Mary','Who','accountant',3);

INSERT INTO Accounts values('Mary','Who','risk analyst',3);

**Part 2**

**Write a python script that is going to load your tables from Part 1 and populate them with data automatically. Use sqlite3 database and make the necessary data type changes (NUMBER(5,0)🡪INTEGER, NUMBER(5,2)🡪REAL).**

**You can use the following sample code for reading the input file and iterating through every line (just be sure that the file is in the local directory or specify the full path):**

fd = open('data\_hw2.txt', 'r')

data = fd.readlines()

fd.close()

for elt in data:

print "Row ", elt

**For this assignment, do not worry about error checking. You could use a dictionary to collect data list for each table first and then use executemany function. However, for this assignment, you can simply write python code that will iterate over all rows in the table and perform separate insert into every table in the schema as it goes along (ignoring errors that will result from attempts at inserting a duplicate key).**

**ANSWER:**

The following script I write tries to read data from the file and put it into three tables accordingly. Running result is also provided below.

scripts

---------------------------------------------------------------------------------------------------------------------

import sqlite3

#read data lines from the file

fd=open('C:/Users/felix/Dropbox/courses/CSC 455/lecture 3/data\_hw2.txt','r')

data = fd.readlines()

fd.close

#connect to the database

conn = sqlite3.connect('csc455.db')

# create table CLIENTS(last,first,address)

Clients='''

CREATE TABLE Clients

( First Varchar2(10),

Last Varchar2(10),

Address Varchar2(30),

CONSTRAINT Clients\_PK

PRIMARY KEY(First,Last)

);'''

# create table LOANS(Lnumber,interest,amount)

Loans='''

CREATE TABLE Loans

( Lnumber Number(2) PRIMARY KEY,

Interest Number(5,2),

Amount Number(10)

);'''

# create accounts(first,last,job,lnumber)

Accounts='''

CREATE TABLE Accounts(

First Varchar2(10),

Last Varchar2(10),

Job Varchar(20),

Lnumber Number(5),

CONSTRAINT AccountsFK

FOREIGN KEY (First,Last)

REFERENCES Clients(First,Last),

CONSTRAINT AccountsFK2

FOREIGN KEY (Lnumber)

REFERENCES Loans(Lnumber),

CONSTRAINT Accounts\_PK

PRIMARY KEY(First,Last,Job,Lnumber)

);'''

#create test Cursor that is trying read data from python database

Select='''select \* from Accounts;'''

#replace existing tables

Clients\_drop='''drop table Clients;'''

Loans\_drop='''drop table Loans;'''

Accounts\_drop='''drop table Accounts;'''

conn.execute(Clients\_drop)

conn.execute(Loans\_drop)

conn.execute(Accounts\_drop)

conn.execute(Clients)

conn.execute(Loans)

conn.execute(Accounts)

#Create strings that contains what is going into the database

data\_Clients=[]

data\_Loans=[]

data\_Accounts=[]

# this whole ITERATION tries to clean the data, transform it into what can be put into the tables and put it into different tables accordingly

for i in range(len(data)):

# clean data

data1=data[i].strip('\n')

data2=data1.split(',')

for m in range(len(data2)):

data2[m] = data2[m].strip(' ')

data2[4]=int(data2[4])

data2[5]=float(data2[5])

data2[6]=int(data2[6])

# put the cleaned data into the strings that is going to be put into tables later. No repeated data is allowed.

if data\_Clients.count([data2[0],data2[1],data2[2]])==0:

data\_Clients.append([data2[0],data2[1],data2[2]])

if data\_Loans.count([data2[4],data2[5],data2[6]])==0:

data\_Loans.append([data2[4],data2[5],data2[6]])

if data\_Accounts.count([data2[0],data2[1],data2[2],data2[3]])==0:

data\_Accounts.append([data2[0],data2[1],data2[2],data2[3]])

# put the strings into the tables

conn.executemany("INSERT INTO Clients VALUES(?,?,?)",data\_Clients)

conn.executemany("INSERT INTO Loans VALUES(?,?,?)",data\_Loans)

conn.executemany("INSERT INTO Accounts VALUES(?,?,?,?)",data\_Accounts)

conn.commit()

# test the script and see what's in the tables.

for row in conn.execute("select \* from Clients"):

print row

for row in conn.execute("select \* from Loans"):

print row

for row in conn.execute("select \* from Accounts"):

print row

---------------------------------------------------------------------------------------------------------------------

result

---------------------------------------------------------------------------------------------------------------------

(u'John', u'Smith', u'111 N. Wabash Avenue')

(u'Jane', u'Doe', u'243 S. Wabash Avenue')

(u'Mike', u'Jackson', u'1 Michigan Avenue')

(u'Mary', u'Who', u'NULL')

(5, 1, 200)

(1, 5, 10000)

(2, 11.8, 7000)

(7, 10, 100)

(8, 20, 25000)

(4, 3.3, 100000)

(3, 2, 12000)

(u'John', u'Smith', u'111 N. Wabash Avenue', u'plumber')

(u'John', u'Smith', u'111 N. Wabash Avenue', u'bouncer')

(u'Jane', u'Doe', u'243 S. Wabash Avenue', u'waitress')

(u'Jane', u'Doe', u'243 S. Wabash Avenue', u'accountant')

(u'Mike', u'Jackson', u'1 Michigan Avenue', u'accountant')

(u'Mary', u'Who', u'NULL', u'accountant')

(u'Mary', u'Who', u'NULL', u'risk analyst')

---------------------------------------------------------------------------------------------------------------------

**Part 3**

**You were hired to do some data analysis for a local zoo. Below is the data table, including the necessary constraints and all the insert statements to populate the database.**

**-- Drop all the tables to clean up**

**DROP TABLE Animal;**

**-- ACategory: Animal category 'common', 'rare', 'exotic'. May be NULL**

**-- TimeToFeed: Time it takes to feed the animal (hours)**

**CREATE TABLE Animal**

**(**

**AID NUMBER(3, 0),**

**AName VARCHAR2(30) NOT NULL,**

**ACategory VARCHAR2(18),**

**TimeToFeed NUMBER(4,2),**

**CONSTRAINT Animal\_PK**

**PRIMARY KEY(AID)**

**);**

**INSERT INTO Animal VALUES(1, 'Galapagos Penguin', 'exotic', 0.5);**

**INSERT INTO Animal VALUES(2, 'Emperor Penguin', 'rare', 0.75);**

**INSERT INTO Animal VALUES(3, 'Sri Lankan sloth bear', 'exotic', 2.5);**

**INSERT INTO Animal VALUES(4, 'Grizzly bear', 'common', 3.0);**

**INSERT INTO Animal VALUES(5, 'Giant Panda bear', 'exotic', 1.5);**

**INSERT INTO Animal VALUES(6, 'Florida black bear', 'rare', 1.75);**

**INSERT INTO Animal VALUES(7, 'Siberian tiger', 'rare', 3.5);**

**INSERT INTO Animal VALUES(8, 'Bengal tiger', 'common', 2.75);**

**INSERT INTO Animal VALUES(9, 'South China tiger', 'exotic', 2.25);**

**INSERT INTO Animal VALUES(10, 'Alpaca', 'common', 0.25);**

**INSERT INTO Animal VALUES(11, 'Llama', NULL, 3.5);**

**Since none of the managers in the zoo know SQL, it is up to you to write the queries to answer the following list of questions.**

1. **Find all the animals (their names) that take less than 2 hours to feed.**

select aname,timetofeed from Animal

where timetofeed < 2.0;

1. **Find all the rare animals and sort the query output by feeding time (any direction)**

select aname,timetofeed from Animal

where acategory = 'rare'

order by timetofeed;

1. **Find the animal names and categories for the animals that are related to a bear (hint: remember the LIKE operator)**

select \* from Animal

where aname like '%bear%';

1. **Return the listings for all animals whose rarity is not available in the database**

select \* from Animal

where acategory is null;

1. **Find the rarity rating of all animals that require between 1 and 2.2 hours to be fed**

select acategory from Animal

where timetofeed >= 1.0 and timetofeed <= 2.2;

1. **Find the names of the animals that are related to the tiger and are not common**

select aname from Animal

where aname like '%tiger%' AND acategory != 'common';

1. **Find the minimum and maximum feeding time amongst all the animals in the zoo**

Select max(timetofeed) from animal;

Select min(timetofeed) from animal;

1. **Find the average feeding time for the rare animals**

Select avg(timetofeed) from animal;

Be sure that your name and “Assignment 2” appear at the top of your submitted file.