Christian Craig
DSC 450 – Hw2

DSC 450: Database Processing for Large-Scale Analytics
Assignment Module 2

Part 1

1) Using your logical schema from previous assignment (Part 2-a), write the necessary SQL DDL script to create the tables. Be sure to specify every primary key and every foreign key. Make reasonable assumptions regarding the attribute domains (for example, setting every column to *VARCHAR2(100)* is not reasonable).

   Screenshots for 1) and 2) within 2)

2) Write SQL INSERT statements to populate your database from Part 2-a with the following data (NOTE: remember that strings would need to use single quotes, e.g., 'Asimov'). Be sure to verify that your statements worked correctly and loaded the data in Oracle (see tutorial on how to connect to Oracle using SQLDeveloper).

   a) (King, Stephen, 2, September 9 1947)
   b) (Asimov, Isaac, 4, January 2 1921)
   c) (Verne, Jules, 7, February 8 1828)
   d) (Rowling, Joanne, 37, July 31 1965)

   e) (Bloomsbury Publishing, 17, London Borough of Camden)
   f) (Arthur A. Levine Books, 18, New York City)

   g) (1111-111, Databases from outer space, 17)
   h) (2222-222, Revenge of SQL, 17)
   i) (3333-333, The night of the living databases, 18)

   j) (2, 1111-111, 1)
   k) (4, 1111-111, 2)
   l) (4, 2222-222, 2)
   m) (7, 2222-222, 1)
   n) (37, 3333-333, 1)
   o) (2, 3333-333, 2)

Christian Craig
DSC 450 – Hw2

Worksheet | Query Builder

```
DROP TABLE Authors;
DROP TABLE Publishers;
DROP TABLE Books;
DROP TABLE Contribution;

CREATE TABLE Authors
(
  LastName VARCHAR2(20),
  FirstName VARCHAR2(20),
  AID NUMBER(*,0),
  BirthDate DATE, --DD-MON-YYYY

  CONSTRAINT Authors_PK
    PRIMARY KEY (AID)

);
```

Query Result

SQL | All Rows Fetched: 4 in 0.025 seconds

	LASTNAME	FIRSTNAME	AID	BIRTHDATE
1	King	Stephen	2	09-SEP-47
2	Asimov	Isaac	4	02-JAN-21
3	Verne	Jules	7	08-FEB-28
4	Rowling	Joanne	37	31-JUL-65

```
CREATE TABLE Publishers
(
  PubName VARCHAR2(30),
  PubNum NUMBER(*,0),
  PubAddress VARCHAR(30),

  CONSTRAINT Publishers_PK
    PRIMARY Key (PubNum)
);
```

Query Result

SQL | All Rows Fetched: 2 in 0.025 seconds

	PUBNAME	PUBNUM	PUBADDRESS
1	Bloomsbury Publishing	17	London Borough of Camden
2	Arthur A. Levine Books	18	New York City

Christian Craig
DSC 450 – Hw2

```
CREATE TABLE Books
(
  ISBN VARCHAR2(8),
  Title VARCHAR2(50),
  PubNum NUMBER(*,0),

  CONSTRAINT Books_PK
    PRIMARY KEY (ISBN),
  CONSTRAINT Books_FK
    FOREIGN KEY (PubNum)
      REFERENCES Publishers(PubNum)
);
```

Query Result | SQL | All Rows Fetched: 3 in 0.024 seconds

	ISBN	TITLE	PUBNUM
1	1111-111	Databases from outer space	17
2	2222-222	Revenge of SQL	17
3	3333-333	The night of the living databases	18

```
CREATE TABLE Contribution
(
AID NUMBER(*,0),
ISBN VARCHAR2(8),
CONT_NUM NUMBER(*,0),

CONSTRAINT Contribution_PK
   PRIMARY Key (AID,ISBN),

CONSTRAINT Contribution_FK1
   FOREIGN KEY (AID)
     REFERENCES Authors (AID),

CONSTRAINT Contribution_FK2
   FOREIGN KEY (ISBN)
     REFERENCES Books (ISBN)

);
```

Query Result | SQL | All Rows Fetched: 6 in 0.024 se

	AID	ISBN	CONT_NUM
1	2	1111-111	1
2	4	1111-111	2
3	4	2222-222	2
4	7	2222-222	1
5	37	3333-333	1
6	2	3333-333	2

```
SELECT * FROM Authors;
SELECT * FROM Publishers;
SELECT * FROM Books;
SELECT * FROM Contribution;

INSERT INTO Authors VALUES('King', 'Stephen',2, '09-SEP-1947')
INSERT INTO Authors VALUES('Asimov', 'Isaac',4, '02-JAN-1921')
INSERT INTO Authors VALUES('Verne', 'Jules',7, '08-FEB-1828')
INSERT INTO Authors VALUES('Rowling','Joanne',37,'31-JUL-1965')

INSERT INTO Publishers VALUES('Bloomsbury Publishing',17,'London Borough of Camden')
INSERT INTO Publishers VALUES('Arthur A. Levine Books', 18, 'New York City')

INSERT INTO Books VALUES('1111-111', 'Databases from outer space', 17)
INSERT INTO Books VALUES('2222-222', 'Revenge of SQL', 17)
INSERT INTO Books VALUES('3333-333', 'The night of the living databases', 18)

INSERT INTO Contribution VALUES(2,'1111-111', 1)
INSERT INTO Contribution VALUES(4,'1111-111', 2)
INSERT INTO Contribution VALUES(4,'2222-222', 2)
INSERT INTO Contribution VALUES(7, '2222-222', 1)
INSERT INTO Contribution VALUES(37, '3333-333', 1)
INSERT INTO Contribution VALUES(2, '3333-333', 2)
```

Christian Craig
DSC 450 – Hw2

3) Using logical schema from previous assignment (Part 2-b) write the necessary SQL DDL script to create the tables. Be sure to specify every primary key and every foreign key.

```
DROP TABLE Departments;
DROP TABLE Advisors;
DROP TABLE Students;

CREATE TABLE Departments
(
  Dept_Name VARCHAR2(30),
  Chair VARCHAR2(30),
  Endowment VARCHAR2(30),

  CONSTRAINT Departments_PK
    PRIMARY KEY (Dept_Name)

);
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.034 seconds

	DEPT_NAME	CHAIR	ENDOWMENT
1	Business	Assistant Chair	George L. Ruff

```
CREATE TABLE Advisors
(
  AID NUMBER(*,0),
  Adv_Name VARCHAR2(20),
  Address VARCHAR2(50),
  Research_Area VARCHAR2(30),
  Dept_Name VARCHAR2(30),

  CONSTRAINT Advisors_PK
    PRIMARY KEY (AID),

  CONSTRAINT Advisors_FK
    FOREIGN KEY (Dept_Name)
        REFERENCES Departments(Dept_Name)
);
```

cript Output | Query Result

SQL | All Rows Fetched: 1 in 0.025 seconds

	AID	ADV_NAME	ADDRESS	RESEARCH_AREA	DEPT_NAME
1	12	Frank Shaw	2247 N. Halsted S...	Blockchain	Business

Christian Craig
DSC 450 – Hw2

```
CREATE TABLE Students
(
  SID NUMBER(*,0),
  LastName VARCHAR2(20),
  FirstName VARCHAR2(20),
  DOB DATE, --DD-MON-YYYY
  Phone VARCHAR2(12),
  AID NUMBER(*,0),

  CONSTRAINT Students_PK
    PRIMARY KEY (SID),

  CONSTRAINT Students_FK
    FOREIGN KEY (AID)
        REFERENCES Advisors (AID)
);

SELECT * FROM Departments;
SELECT * FROM Advisors;
SELECT * FROM Students;
```

ipt Output × | Query Result ×

SQL | All Rows Fetched: 1 in 0.025 seconds

	SID	LASTNAME	FIRSTNAME	DOB	PHONE	AID
1	145	Smith	John	31-JUL-65	773-426-4554	12

4) Write a python function to validate SQL insert statements. The function will take a string containing a SQL insert statement and print two kinds of messages. 1) “Invalid insert” or 2) Inserting (1, Jane, A+) into Students table. We are doing only a very limited SQL validation, so you only have to check that the insert statement starts with INSERT INTO and that statements ends with ;

   Examples:
   validateInsert('**INSERT INTO Students VALUES (1, Jane, A+);**')
   - Inserting (1, Jane, A+) into Students table

   validateInsert('**INSERT INTO Students VALUES (1, Jane, A+)**')
   - Invalid insert

   validateInsert('**INSERT Students VALUES (1, Jane, A+);**')
   - Invalid insert

   validateInsert('**INSERT INTO Phones VALUES (42, 312-555-1212);**')
   - Inserting (42, 312-555-1212) into Phones table

Christian Craig
DSC 450 – Hw2

```
def validateInsert(stmnt):
    '''
    function that validates sql
    insert statement
    '''
    lst=stmnt.split()
    if lst[0:2]==['INSERT','INTO'] and lst[-1][-1] ==';': #if passes
        print('Inserting ({}) into {} table'.format(stmnt[stmnt.find("(") + 1:stmnt.find(")")], lst[2]))
    else: #if fails
        print('Invalid Insert')
```

```
>>> validateInsert('INSERT INTO Students VALUES (1, Jane, A+);')
Inserting (1, Jane, A+) into Students table
>>> validateInsert('INSERT INTO Students VALUES (1, Jane, A+)')
Invalid Insert
>>> validateInsert('INSERT Students VALUES (1, Jane, A+);')
Invalid Insert
>>> validateInsert('INSERT INTO Phones VALUES (42, 312-555-1212);')
Inserting (42, 312-555-1212) into Phones table
```

Christian Craig
DSC 450 – Hw2

Part 2

Consider a MEETING table that records information about meetings between clients and executives in the company. Each record contains the names of the client and the executive's name as well as the office number, floor and the building. Finally, each record contains the city that the building is in and the date of the meeting. The table is in First Normal Form and the primary key is (Client, Office).
(Date, Client, Office, Floor, Building, City, Executive)

You are given the following functional dependencies:
Building → City
Office → Floor, Building, City
Client → Executive
Client, Office → Date

a. **For the functional dependency Building → City, explain the redundancy problem and possible consequences through an example (you can make up your own building names as you see fit).**

Meeting						
Date	Client	Office	Floor	Building	City	Executive
05/21/2020	A	N234	2	North	Troy	Josh G.
5/22/2020	B	S126	1	South	Detroit	Jaime T.
5/23/2020	C	N234	2	North	Troy	Sarah S.
5/23/2020	B	N244	2	North	Troy	Jamie T.

From the above, we can see that Building→ City causes redundancy. Specifically, we can see that each time Building = North, we see City = Troy and each time we see Building = South, we see city = Detroit.

For this, we can separate the table as shown below:

Meeting					
Date	Client	Office	Floor	Building	Executive
05/21/2020	A	N234	2	North	Josh G.
5/22/2020	B	S126	1	South	Jamie T.
5/23/2020	C	N234	2	North	Sarah S.
5/23/2020	B	N244	2	North	Jamie T.

Building	City
North	Troy
South	Detroit

Now we can see that city is removed and the other table serves the purpose of which Building belongs in which city, adhering to Building → City.

b. Remove any existing partial dependencies and convert the logical schema to the Second Normal Form. Please remember that when performing schema decomposition, you need to denote primary key for every new table as well as the foreign key that will allow us to reconstruct the original data.

Date	Client	Office	Floor	Building	City	Executive

Building → City **Office → Floor, Building, City**
Client → Executive **Client, Office → Date**

M1(*Client*, Executive)
↓
M2(Client, Office, Date, Executive)

M3(Office, Floor, Building, City)

M4(*Client*, Office, Date)
↓
M5(Client, Executive)

2NF

c. Remove any existing transitive dependencies to create a set of logical schemas in Third Normal Form. Again, remember to denote primary keys and foreign keys (including which primary key those foreign keys point to).

Date	Client	Office	Floor	Building	City	Executive

Building → City **Office → Floor, Building, City**
Client → Executive **Client, Office → Date**

M3(Client, Office, Date)

M4(*Client*, Executive) M5(*Office*, Floor, Building)

M6(*Building*, City)

Christian Craig
DSC 450 – Hw2

Part 3

Consider a table that stores information about students, student name, GPA, honors list and the credits that the student had completed so far.

(First, Last, GPA, Honor, Credits)

You are given the following functional dependencies

First, Last → GPA, Honor, Credits
GPA → Honor

a. Is this schema in Second Normal Form? If not, please state which FDs violate 2NF and decompose the schema accordingly.

The schema is in second normal form. We can conclude that there are no partial functional dependencies.

b. Is this schema in Third Normal Form? If not, please state which FDs violate 3NF and decompose the schema accordingly.

The schema is not in third normal form. There is a transitive function where GPA determines Honor. We need to ensure that our entire schema needs to be complete in this manner.

S1(GPA, Honor)

S2(First, Last, GPA, Credits)