Part-1) Create Table for 3NF by SQL (No Partial Dependencies & No Transition Dependencies)

3NF:

Employee( employeeFirst, employeeLast, employeeAddress)

Jobs(employeeJob, employeeSalary, employeeAssistant)

Employee\_Jobs(employeeFirst, employeeLast, employeeJob)

Note:

employeeFirst, employeeLast are Foreign Keys referencing to Table Employee

employeeJob is Foreign Key referencing to Table Jobs

/\*DROP ALL the Tables \*/

DROP TABLE Employee\_Jobs;

DROP TABLE Employee;

DROP TABLE Jobs;

/\*Create Table for Employee\*/

CREATE TABLE Employee

(

employeeFirst varchar2(15) NOT NULL,

employeeLast varchar2(15) NOT NULL,

employeeAddress varchar2(50),

CONSTRAINT Employee\_PK

PRIMARY KEY (employeeFirst,employeeLast)

);

/\*Create Table for Jobs\*/

CREATE TABLE Jobs

(

employeeJob varchar2(15) NOT NULL,

employeeSalary varchar2(5),

employeeAssistant varchar2(3),

CONSTRAINT Jobs\_PK

PRIMARY KEY (employeeJob)

);

/\*Create Table for Emloyee & Jobs\*/

CREATE TABLE Employee\_Jobs

(

employeeFirst varchar2(15) NOT NULL,

employeeLast varchar2(15) NOT NULL,

employeeJob varchar2(15) NOT NULL,

CONSTRAINT Employee\_Jobs\_PK

PRIMARY KEY (employeeFirst, employeeLast, employeeJob),

CONSTRAINT Employee\_Jobs\_FK1

FOREIGN KEY (employeeFirst, employeeLast)

REFERENCES Employee (employeeFirst,employeeLast),

CONSTRAINT Employee\_Jobs\_FK3

FOREIGN KEY (employeeJob)

REFERENCES Jobs (employeeJob)

);

Part-1) Load data provided in the text file data\_hw2.txt by using Python Code:

*#Open the provided txt file*

fd=open("data\_hw2.txt")

lines=fd.readlines()

stringLst=[]

*#Reading and Cleaning the data line by line*

**for** l **in** lines:

a=l.strip().split(', ') *# Get rid all the spaces*

stringLst.append(a)

fd.close()

*#View the data read from the data\_hw2.txt*

stringLst

Out[2]:

[['John', 'Smith', '111 N. Wabash Avenue', 'plumber', '40K', 'NULL'],

['John', 'Smith', '111 N. Wabash Avenue', 'bouncer', '35K', 'NULL'],

['Jane', 'Doe', '243 S. Wabash Avenue', 'waitress', '50K', 'NULL'],

['Jane', 'Doe', '243 S. Wabash Avenue', 'accountant', '42K', 'Yes'],

['Jane', 'Doe', '243 S. Wabash Avenue', 'bouncer', '35K', 'NULL'],

['Mike', 'Jackson', '1 Michigan Avenue', 'accountant', '42K', 'Yes'],

['Mike', 'Jackson', '1 Michigan Avenue', 'plumber', '40K', 'NULL'],

['Mary', 'Who', '20 S. Michigan Avenue', 'accountant', '42K', 'Yes'],

['Mary', 'Who', '20 S. Michigan Avenue', 'risk analyst', '80K', 'Yes']]

Part-1) Generate INSERT OR IGNORE Statement by using Python Function:

*#Function to Generate the SQL Insert OR IGNORE Statement*

**def** generateInsert(tableName,inputList):

newList=[]

*#Check and turn all the numerical value to integer*

**for** item **in** inputList:

**if** item.isdigit()== **True**:

item=int(item)

newList.append(item)

**else**:

newList.append(item)

*#Subset the input source file to generate Insert statement for Table 'Employee','Jobs','Employee\_Jobs'*

**if** tableName=='Employee':

**return** "INSERT OR IGNORE INTO **{}** VALUES ('**{}**','**{}**','**{}**');".format(tableName, newList[0],newList[1],newList[2])

**elif** tableName=='Jobs':

*#Convert the 'Null' value to compatible to SQL code by using '''NULL'''*

**if** str(newList[5])=="NULL":

**return** "INSERT OR IGNORE INTO **%s** VALUES ('**%s**','**%s**',**%s**);"%(tableName, newList[3],newList[4],'''NULL''')

**else**:

**return** "INSERT OR IGNORE INTO **{}** VALUES ('**{}**','**{}**','**{}**');".format(tableName, newList[3],newList[4],newList[5])

**elif** tableName=='Employee\_Jobs':

**return** "INSERT OR IGNORE INTO **{}** VALUES ('**{}**','**{}**','**{}**');".format(tableName, newList[0],newList[1],newList[3])

*#Display the INSERT OR IGNORE statements for Table "Employee"*

**for** l **in** stringLst:

l=generateInsert("Employee", l)

print(l)

INSERT OR IGNORE INTO Employee VALUES ('John','Smith','111 N. Wabash Avenue');

INSERT OR IGNORE INTO Employee VALUES ('John','Smith','111 N. Wabash Avenue');

INSERT OR IGNORE INTO Employee VALUES ('Jane','Doe','243 S. Wabash Avenue');

INSERT OR IGNORE INTO Employee VALUES ('Jane','Doe','243 S. Wabash Avenue');

INSERT OR IGNORE INTO Employee VALUES ('Jane','Doe','243 S. Wabash Avenue');

INSERT OR IGNORE INTO Employee VALUES ('Mike','Jackson','1 Michigan Avenue');

INSERT OR IGNORE INTO Employee VALUES ('Mike','Jackson','1 Michigan Avenue');

INSERT OR IGNORE INTO Employee VALUES ('Mary','Who','20 S. Michigan Avenue');

INSERT OR IGNORE INTO Employee VALUES ('Mary','Who','20 S. Michigan Avenue');

In [20]:

*#Display the INSERT OR IGNORE statements for Table "Jobs"*

**for** l **in** stringLst:

l=generateInsert("Jobs", l)

print(l)

INSERT OR IGNORE INTO Jobs VALUES ('plumber','40K',NULL);

INSERT OR IGNORE INTO Jobs VALUES ('bouncer','35K',NULL);

INSERT OR IGNORE INTO Jobs VALUES ('waitress','50K',NULL);

INSERT OR IGNORE INTO Jobs VALUES ('accountant','42K','Yes');

INSERT OR IGNORE INTO Jobs VALUES ('bouncer','35K',NULL);

INSERT OR IGNORE INTO Jobs VALUES ('accountant','42K','Yes');

INSERT OR IGNORE INTO Jobs VALUES ('plumber','40K',NULL);

INSERT OR IGNORE INTO Jobs VALUES ('accountant','42K','Yes');

INSERT OR IGNORE INTO Jobs VALUES ('risk analyst','80K','Yes');

In [21]:

*#Display the INSERT OR IGNORE statements for Table "Employee\_Jobs"*

**for** l **in** stringLst:

l=generateInsert("Employee\_Jobs", l)

print(l)

INSERT OR IGNORE INTO Employee\_Jobs VALUES ('John','Smith','plumber');

INSERT OR IGNORE INTO Employee\_Jobs VALUES ('John','Smith','bouncer');

INSERT OR IGNORE INTO Employee\_Jobs VALUES ('Jane','Doe','waitress');

INSERT OR IGNORE INTO Employee\_Jobs VALUES ('Jane','Doe','accountant');

INSERT OR IGNORE INTO Employee\_Jobs VALUES ('Jane','Doe','bouncer');

INSERT OR IGNORE INTO Employee\_Jobs VALUES ('Mike','Jackson','accountant');

INSERT OR IGNORE INTO Employee\_Jobs VALUES ('Mike','Jackson','plumber');

INSERT OR IGNORE INTO Employee\_Jobs VALUES ('Mary','Who','accountant');

INSERT OR IGNORE INTO Employee\_Jobs VALUES ('Mary','Who','risk analyst');

Part2) Write Python Script to load the data:

*#Step 1: Load all the data from provided ‘data\_hw2.txt’*

*#Open the provided txt file*

fd=open("data\_hw2.txt")

lines=fd.readlines()

stringLst=[]

*#Reading and Cleaning the data line by line*

**for** l **in** lines:

a=l.strip().split(', ') *# Get rid all the spaces*

stringLst.append(a)

fd.close()

*#View the data read from the data\_hw2.txt*

stringLst

Out[2]:

[['John', 'Smith', '111 N. Wabash Avenue', 'plumber', '40K', 'NULL'],

['John', 'Smith', '111 N. Wabash Avenue', 'bouncer', '35K', 'NULL'],

['Jane', 'Doe', '243 S. Wabash Avenue', 'waitress', '50K', 'NULL'],

['Jane', 'Doe', '243 S. Wabash Avenue', 'accountant', '42K', 'Yes'],

['Jane', 'Doe', '243 S. Wabash Avenue', 'bouncer', '35K', 'NULL'],

['Mike', 'Jackson', '1 Michigan Avenue', 'accountant', '42K', 'Yes'],

['Mike', 'Jackson', '1 Michigan Avenue', 'plumber', '40K', 'NULL'],

['Mary', 'Who', '20 S. Michigan Avenue', 'accountant', '42K', 'Yes'],

['Mary', 'Who', '20 S. Michigan Avenue', 'risk analyst', '80K', 'Yes']]

Part2) Write Python Script to CREATE Table and Populate the data into the Table:

*#Step2: Create Table 'Employee' Script*

Em = '''CREATE TABLE Employee

(

employeeFirst varchar(15) NOT NULL,

employeeLast varchar(15) NOT NULL,

employeeAddress varchar2(50),

CONSTRAINT Employee\_PK

PRIMARY KEY (employeeFirst,employeeLast)

);'''

*#Step3: Drop the Table 'Employee'*

c.execute("DROP TABLE Employee")

*#Step4: Populate the Table 'Employee'*

c.execute(Em)

Out[]:

<sqlite3.Cursor at 0x10eb0d5e0>

*#Step5:Collecting all the INSERT Statement to a List which is called "filterList"*

filterList=[]

**for** l **in** stringLst:

l=generateInsert("Employee", l)

filterList.append(l)

*#Own Reference: Filtering out all the DUPLICATED INSERT Statement*

*#insertList=[]*

*#for i in range(len(filterList)):*

*#print(i)*

*# if filterList[i] in insertList:*

*# pass*

*# else:*

*# insertList.append(filterList[i])*

*#Step6: populate the tables with the provided data*

**for** l **in** filterList:

*#Execuate the INSERT statement*

c.execute(l)

In [23]:

*#Step7: Double Check and Query the 'Employee' Table*

c.execute("select \* from Employee;").fetchall()

Out[23]:

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

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

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

('Mary', 'Who', '20 S. Michigan Avenue')]

*#Step8: Create Table 'Jobs' Script*

Jo='''CREATE TABLE Jobs

(

employeeJob varchar(15) NOT NULL,

employeeSalary varchar(5),

employeeAssistant varchar2(3),

CONSTRAINT Jobs\_PK

PRIMARY KEY (employeeJob)

);'''

*#Step9: DROP the 'Jobs' TABLE*

c.execute("DROP TABLE Jobs")

*#Setp10: Populate the Table 'Jobs'*

c.execute(Jo)

Out[]:

<sqlite3.Cursor at 0x10eb0d5e0>

In []:

*#Step11: Collecting all the INSERT Statement to a List which is called "filterList"*

filterList=[]

**for** l **in** stringLst:

l=generateInsert("Jobs", l)

filterList.append(l)

*#Filtering out all the DUPLICATED INSERT Statement*

*#insertList=[]*

*#for i in range(len(filterList)):*

*#print(i)*

*# if filterList[i] in insertList:*

*# pass*

*# else:*

*# insertList.append(filterList[i])*

*#Step12: populate the tables with the provided data*

**for** l **in** filterList:

*#Execuate the INSERT statement*

c.execute(l)

In [29]:

*#Step13: Double Check and Query the 'Jobs' Table*

c.execute("select \* from Jobs;").fetchall()

Out[29]:

[('plumber', '40K', None),

('bouncer', '35K', None),

('waitress', '50K', None),

('accountant', '42K', 'Yes'),

('risk analyst', '80K', 'Yes')]

In [30]:

*#Step14: Double check the listings where employeeAssistant is NULL*

c.execute("select \* from Jobs where employeeAssistant is NULL").fetchall()

Out[30]:

[('plumber', '40K', None), ('bouncer', '35K', None), ('waitress', '50K', None)]

*#Step15: Create Table 'Employee\_Jobs' Script*

EmJo='''CREATE TABLE Employee\_Jobs

(

employeeFirst varchar(15) NOT NULL,

employeeLast varchar(15) NOT NULL,

employeeJob varchar(15) NOT NULL,

CONSTRAINT Employee\_Jobs\_PK

PRIMARY KEY (employeeFirst, employeeLast, employeeJob),

CONSTRAINT Employee\_Jobs\_FK1

FOREIGN KEY (employeeFirst, employeeLast)

REFERENCES Employee (employeeFirst,employeeLast),

CONSTRAINT Employee\_Jobs\_FK3

FOREIGN KEY (employeeJob)

REFERENCES Jobs (employeeJob)

);'''

*#Step16: Drop the 'Employee\_jobs' Table*

c.execute("DROP TABLE Employee\_Jobs")

*#Step17: Populate the Table 'Employee\_Hobs'*

c.execute(EmJo)

Out[]:

<sqlite3.Cursor at 0x10eb0d5e0>

*#Step17: Collecting all the INSERT Statement to a List which is called "filterList"*

filterList=[]

**for** l **in** stringLst:

l=generateInsert("Employee\_Jobs", l)

filterList.append(l)

*#Filtering out all the DUPLICATED INSERT Statement*

*#insertList=[]*

*#for i in range(len(filterList)):*

*#print(i)*

*# if filterList[i] in insertList:*

*# pass*

*# else:*

*# insertList.append(filterList[i])*

*#Step18: populate the tables with the provided data*

**for** l **in** filterList:

*#Execuate the INSERT statement*

c.execute(l)

In [37]:

*#Step19: Double Check and Query the 'Employee\_Jobs' Table*

c.execute("SELECT \* from Employee\_Jobs;").fetchall()

Out[37]:

[('John', 'Smith', 'plumber'),

('John', 'Smith', 'bouncer'),

('Jane', 'Doe', 'waitress'),

('Jane', 'Doe', 'accountant'),

('Jane', 'Doe', 'bouncer'),

('Mike', 'Jackson', 'accountant'),

('Mike', 'Jackson', 'plumber'),

('Mary', 'Who', 'accountant'),

('Mary', 'Who', 'risk analyst')]

Part 3) Write the queries to answer questions:

/\*\*\*\*\*\*\*\*\*\*\*Provided Script\*\*\*\*\*\*\*\*\*\*\*/

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);

select \* from Animal;

Answers:

/\*Part3-1\*/

SELECT ANAME from Animal

where TIMETOFEED<1.5;

/\*Part3-2\*/

SELECT \* from Animal

where ACATEGORY='rare'

order by TIMETOFEED;

/\*Part3-3\*/

SELECT ANAME, ACATEGORY from Animal

where ANAME LIKE '%bear%';

/\*Part3-4\*/

SELECT \* from Animal

where ACATEGORY is NULL;

/\*Part3-5\*/

SELECT ACATEGORY from Animal

where TIMETOFEED BETWEEN 1 AND 2.5;

/\*Part3-6\*/

SELECT ANAME from Animal

where ANAME LIKE '%tiger%' AND ACATEGORY != 'common';

/\*Part3-7\*/

SELECT ANAME FROM Animal

where ANAME NOT LIKE '%tiger%';

/\*Part3-8\*/

/\*Minimum\*/

SELECT MIN(TIMETOFEED) from Animal;

/\*Maximum\*/

SELECT MAX(TIMETOFEED) from Animal;

/\*Part3-9\*/

/\*Round Average Result upto 4 Decimal place\*/

SELECT ROUND(AVG(TIMETOFEED),4) from Animal;