

DSC 450: Database Processing for Large-Scale Analytics

Assignment Module 1

NAME: SYED NOOR RAZI ALI

STUDENT ID: 2070326

Part 1

- A) Write a python function that will accept a name of a text file containing multiple rows with comma separated numbers (for example “5, 6, 12, 56, 1”) and return the average of these numbers. Your function should compute and average all of the numbers in a file.
Note that your function should return the result of the computation (i.e., return res not print(res)).

```
In [33]: #Defining a function
def AVGFunction():
    data = [] #Defining an Empty List
    with open('C:/Users/razia/OneDrive/Documents/avenumbers.txt') as f: #Opening the text file
        for line in f: # for Loop Iterates for each row in the file
            fields = line.split(',') #numbers are split by a comma in thr text file
            rowdata = map(float,fields)
            data.extend(rowdata) #Add them to the List
        avg = (sum(data)/len(data)); #Find the average
        return avg; #Return the average

if __name__ == '__main__': #stand alone module run by the user.
    #Function Calling
    avg = AVGFunction();
    #print(average) upto 2 decimal places
    print("Average = ", "{0:.2f}".format(average))

Average = 23.24
```

- B) Write a python function that is going to generate and return a SQL INSERT statement given a table name and value list as parameters. We have not covered SQL yet, so you don't need to run these insert statements, just get your python code to return the required string. Do not hardcode the output.

For example,

print(generateInsert('Students', ['1', 'Jane', 'B+'])) should print
INSERT INTO Students VALUES (1, Jane, B+);

All statements start from **INSERT INTO**, followed by the name of the table, followed by **VALUES** and the comma-separated list of supplied values inside parenthesis, terminated by a semi-colon
Make sure that your function returns the string rather than prints it.

If you are interested in additional challenge, modify your function to instead return (this modification is not required):

INSERT INTO Students VALUES (1, 'Jane', 'B+');
(i.e., put quotes around strings, but not around numbers).

Another example:

`generateInsert('Phones', ['42', '312-556-1212'])` should return
`INSERT INTO Phones VALUES (42, '312-556-1212');`

Both of the example calls (with different number of values) should work correctly.

```
Average = 23.24

In [43]: def SQLtable(table,Values):
          S = ('',''.join(Values))
          return 'INSERT INTO {} VALUES ({});'.format(table,S)
          print(SQLtable('TABLENAME',['1','JANE','A+']))
          print(SQLtable('TABLENAME 2',['42','8328177395']))

INSERT INTO TABLENAME VALUES (1,JANE,A+);
INSERT INTO TABLENAME 2 VALUES (42,8328177395);
```

Part 2

- a) Define a relational schema with underlined (primary) keys and arrows connecting foreign keys and primary keys for a database containing the following information.
- **Authors** have LastName, FirstName, ID, and Birthdate (identified by ID)
 - **Publishers** have Name, PubNumber, Address (identified by PubNumber)
 - **Books** have ISBN, Title, Publisher (each book has a publisher and is identified by its ISBN).
 - Authors **Write** Books; since many authors can co-author a book, we need to know the relative contribution of the author to a book, signified by their position in the author list (i.e. 1, 2, 3, etc.).

SOLUTION:

AUTHOR - LastName, FirstName, ID, and Birthdate

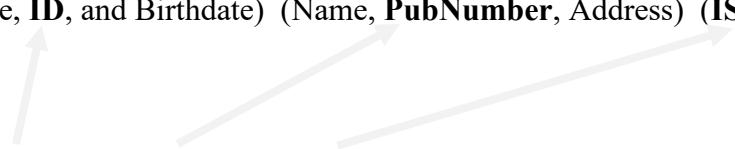
PUBLISHER - Name, PubNumber, Address

BOOKS - ISBN, Title, Publisher

Here ID from Author , PUBNUMBER from publisher and ISBN from Books are the **Primary keys**

We can create a joining table with ID, PUBNUMBER, ISBN and AUTHOR RANK where ID, PUBNUMBER, ISBN can be the **primary and foreign keys** joining the tables.

(LastName, FirstName, **ID**, and Birthdate) (Name, **PubNumber**, Address) (**ISBN**, Title, Publisher)



(ID, PUBNUMBER, ISBN ,AUTHOR RANK)

Create sample data for at least 2 books, at least 1 publisher and at least 2 authors (who co-authored the same book).

Author

LastName	FirstName	ID	BirthDate
Shakespeare	William	1	12/23/1832
Markaram	Charlie	2	09/15/1987
Jhones	David	3	07/23/1838

Publisher

Name	PubNumber	Address
Black Label	548213	London
Sky Feathers	985641	Manchester

Books

ISBN	Title	Publisher
6548756	Merchant of Venice	Black Label
1245879	Royalty vs Poverty	Sky Feathers

New Table

ID	PubNumber	ISBN	AuthorRank
1	548213	6548756	1
3	548213	6548756	2

b) Define a relational schema for students, student advisors, and advisor departments

- **Students** have StudentID, First Name, Last Name, DOB, Telephone and a reference to their advisor
- **Advisors** have ID, Name, Address, Research Area, and a reference link to their Department
- **Departments** have Name, Chair, Endowment (identified by Name)

SOLUTION:

(**StudentID**, First Name, Last Name, DOB, Telephone, Advisor)

(**ID**, Name, Address, Research Area Department,) (**Name**, Chair, Endowment,Name)

(STUDENTID, ID, DEPT_NAME)

