Semi-structure Data Processing: Transforming XML data to CSV format

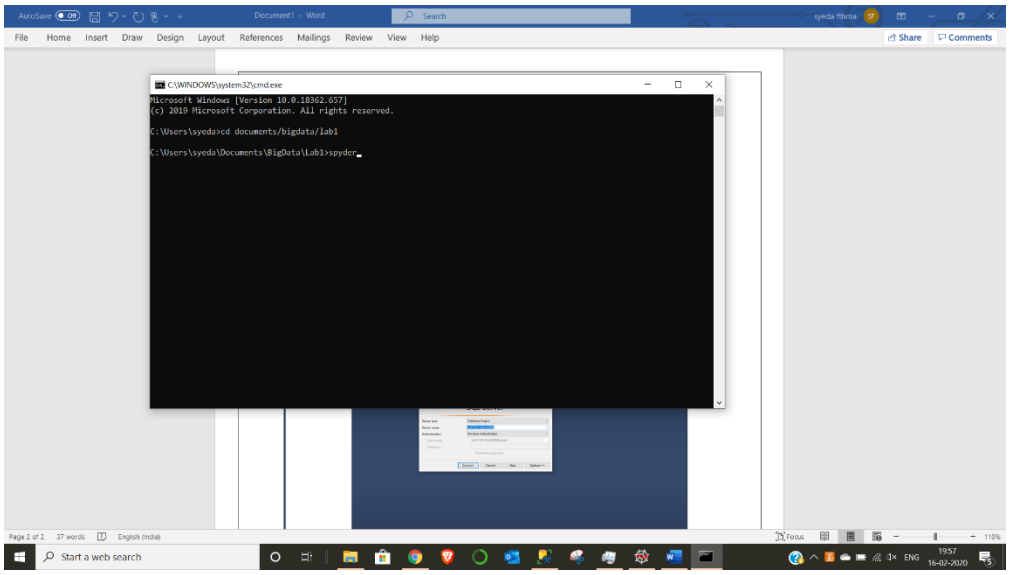
Fathima Syeda

CSU ID-2790024

Platform Setup:

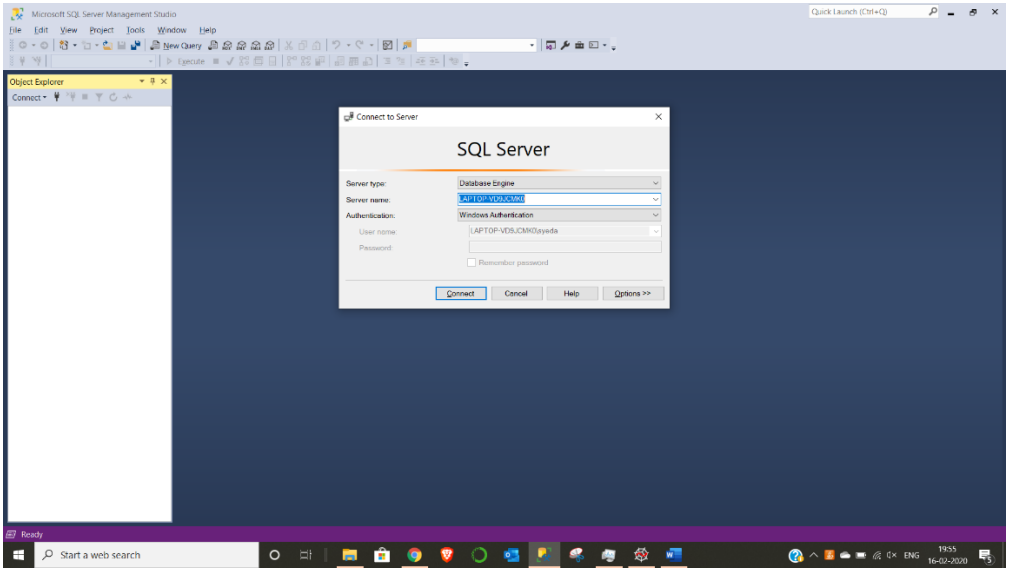
This lab was done using Microsoft SQL Server Management Studio and Python.

1) Spyder IDE was used to write the python code. In order to open Spyder you go to the command prompt and go to the directory you wish to save your source code in and then type Spyder to launch the Spyder IDE.



Once Spyder opens, you start typing your python source code.

2) The server can be started by opening the Microsoft SQL Server Management Studio application.

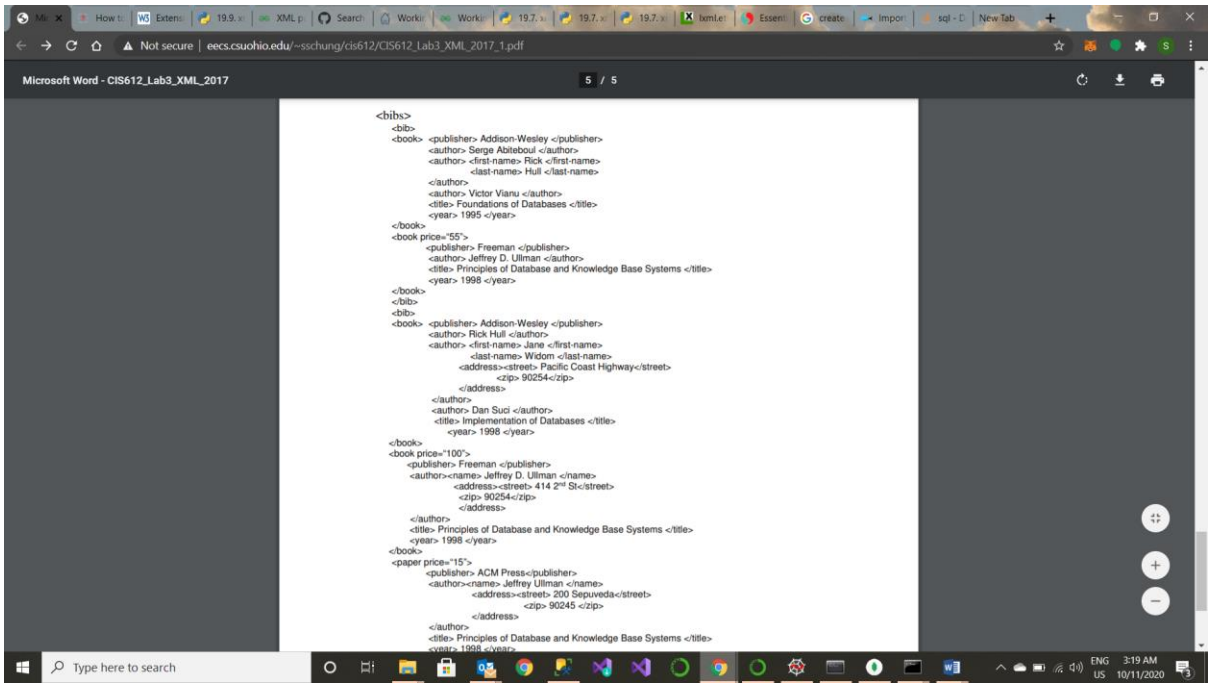


Once it opens you may make note of your server's name i.e. "HASNAIN2020 " (which will be used in the code)and press connect to connect to it.

Program Outline:

The program makes use of python's xml, pyodbc libraries.

The given xml file is parsed using python's xml parser and the using the ElementTree library we access the text in the semi-structured data.



```
<bibs>
  <bib>
  <book>  <publisher> Addison-Wesley </publisher>
          <author> Serge Abiteboul </author>
          <author> <first-name> Rick </first-name>
                   <last-name> Hull </last-name>
          </author>
          <author> Victor Vianu </author>
          <title> Foundations of Databases </title>
          <year> 1995 </year>
  </book>
  <book price="55">
          <publisher> Freeman </publisher>
          <author> Jeffrey D. Ullman </author>
          <title> Principles of Database and Knowledge Base Systems </title>
          <year> 1998 </year>
  </book>
  </bib>
  <bib>
  <book>  <publisher> Addison-Wesley </publisher>
          <author> Rick Hull </author>
          <author> <first-name> Jane </first-name>
                   <last-name> Widom </last-name>
                <address><street> Pacific Coast Highway</street>
                         <zip> 90254</zip>
                </address>
          </author>
          <author> Dan Suci </author>
          <title> Implementation of Databases </title>
          <year> 1998 </year>
  </book>
  <book price="100">
      <publisher> Freeman </publisher>
      <author><name> Jeffrey D. Ullman </name>
              <address><street> 414 2nd St</street>
                <zip> 90254</zip>
                </address>
      </author>
      <title> Principles of Database and Knowledge Base Systems </title>
      <year> 1998 </year>
  </book>
  <paper price="15">
          <publisher> ACM Press</publisher>
          <author><name> Jeffrey Ullman </name>
                  <address><street> 200 Sepuveda</street>
                           <zip> 90245 </zip>
                  </address>
          </author>
          <title> Principles of Database and Knowledge Base Systems </title>
          <year> 1998 </year>
```

Once the semistructured data has been parsed and the root is found , we study the data and decide on a relational schema that the xml data should be converted to.

The semistructured data has <bibs> which has 2 <bib> . Each of the <bib> have multiple <book> and <paper> tags. Each of the <book> and <paper> tag have <title>, <publisher>,<author>, <year> and @price attribute. The <author> tag is further nested to <first-name> <last-name> or <name> tags and an <address>. The address tag has <zip>, <street> in it.

The xml structure would be put into a relational database in 4 Relational tables –Bibs , Bibliography, Author and Address tables.

The **Bibs table** consists of bib_id and bib_name columns where bib_id is the primary key.

The **Bibliography table** looks like this

```
CREATE TABLE dbo.BIBLIOGRAPHY (
                                        ITEM_ID INT PRIMARY KEY,
                                        ITEM_TYPE VARCHAR(10),
                                        PRICE INT,
                                        PUBLISHER VARCHAR(45),
                                        TITLE VARCHAR(45),
                                        YEAR INT,
                                        BIB_ID INT NOT NULL,
    CONSTRAINT BIBLIOGRAPHY_fk_group FOREIGN KEY (BIB_ID) REFERENCES dbo.BIBS(BIB_ID)

                                        );.
```

The **Author table** looks like this:

```
CREATE TABLE dbo.AUTHOR (
                                        AUTH_ID INT PRIMARY KEY,
                                        AUTH_NAME VARCHAR(60),
                                        ITEM_ID INT NOT NULL,
    CONSTRAINT AUTHOR_fk_group FOREIGN KEY (ITEM_ID) REFERENCES
dbo.BIBLIOGRAPHY(ITEM_ID)

                                        );
```

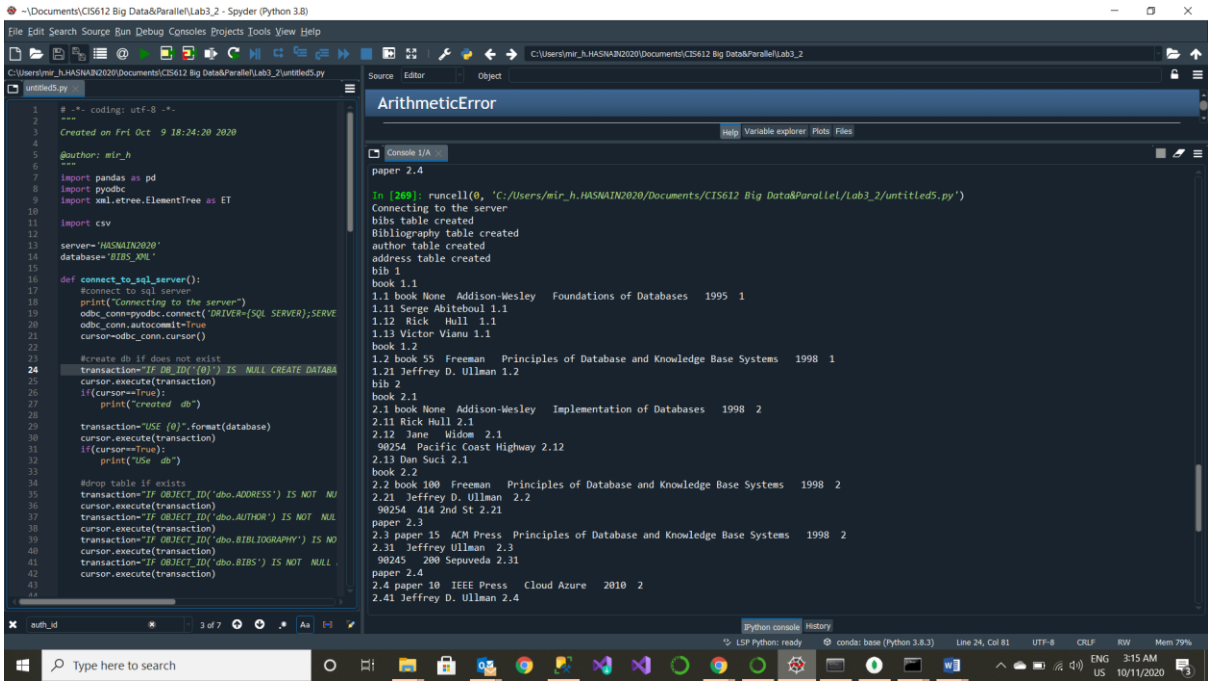
And the **Address table** looks like this:

```
CREATE TABLE dbo.ADDRESS (
                                        ZIP INT,
                                        STREET VARCHAR(60),
                                        AUTH_ID INT NOT NULL,
    CONSTRAINT ADDRESS_fk_group FOREIGN KEY (AUTH_ID) REFERENCES dbo.AUTHOR(AUTH_ID)
                                    );
```

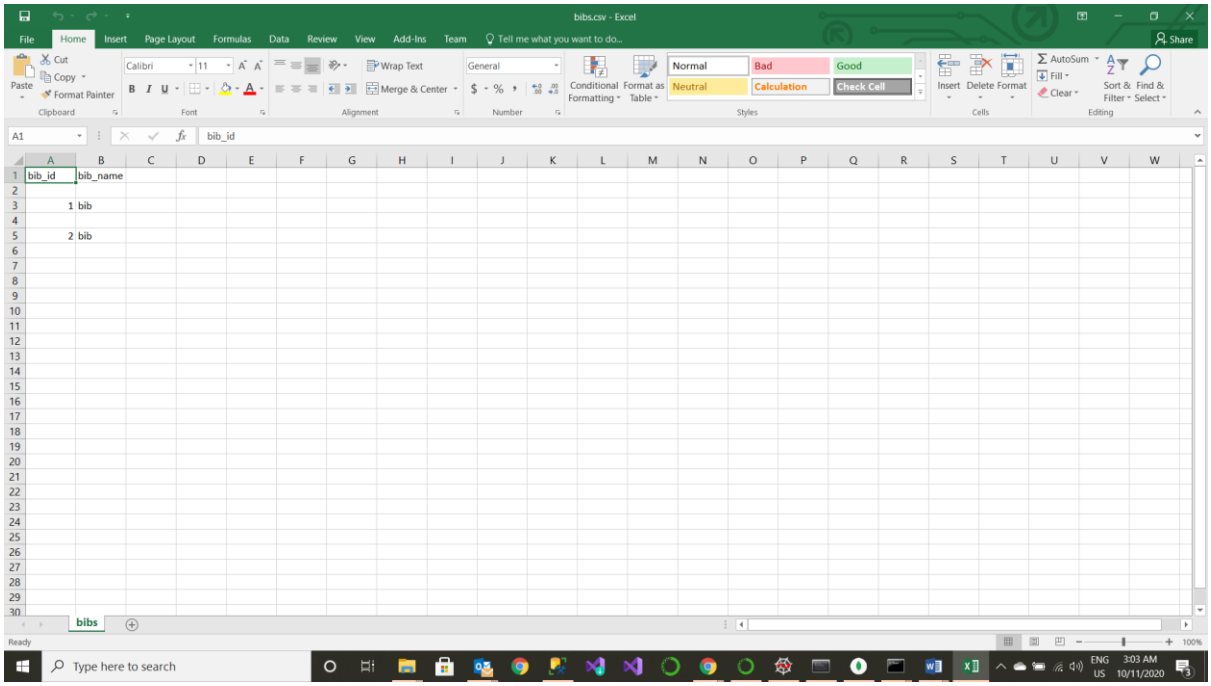
The above tables are automatically created in the python script after parsing the xml document.

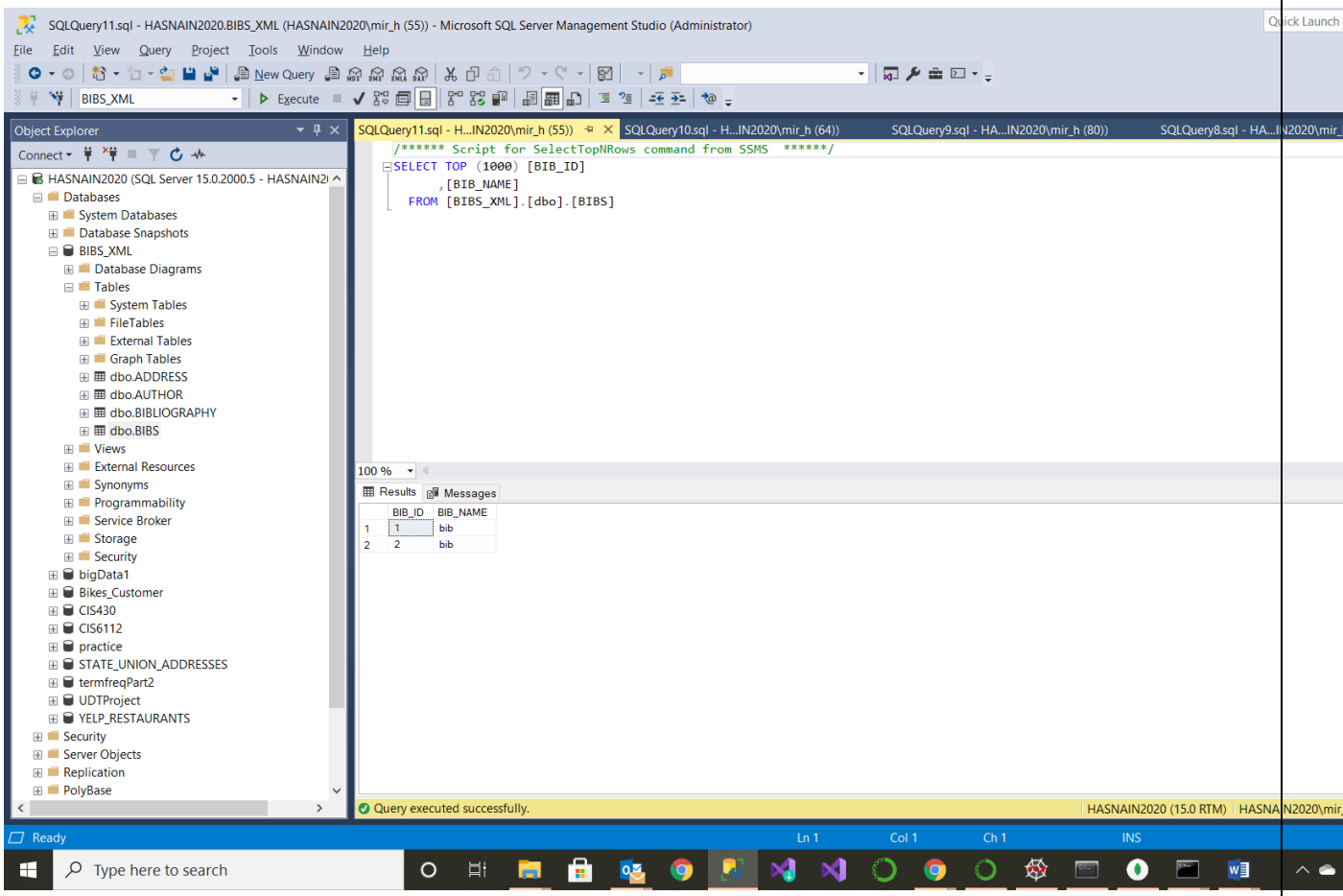
The database housing these tables is the BIBS_XML database.

Once the tables are created the xml data parsed is written into csv files and from there it is inserted into the SQL table.

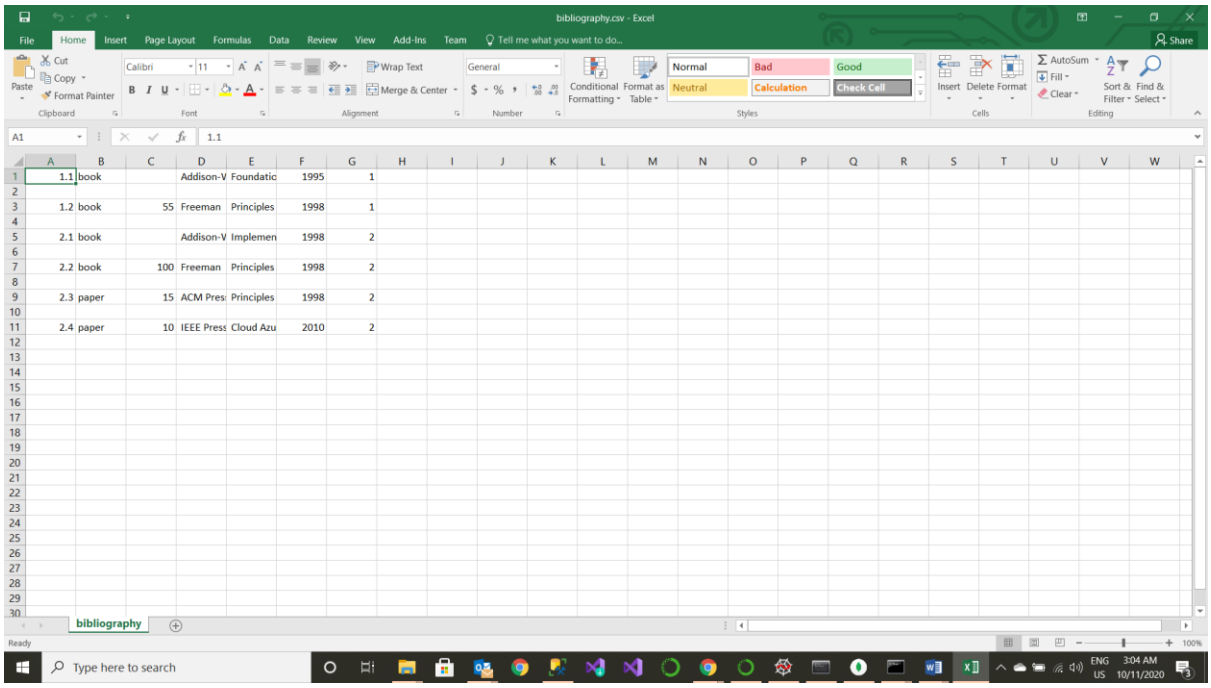


The bibs data consists of 2 entries into 2 columns of bib_id and bib_name.

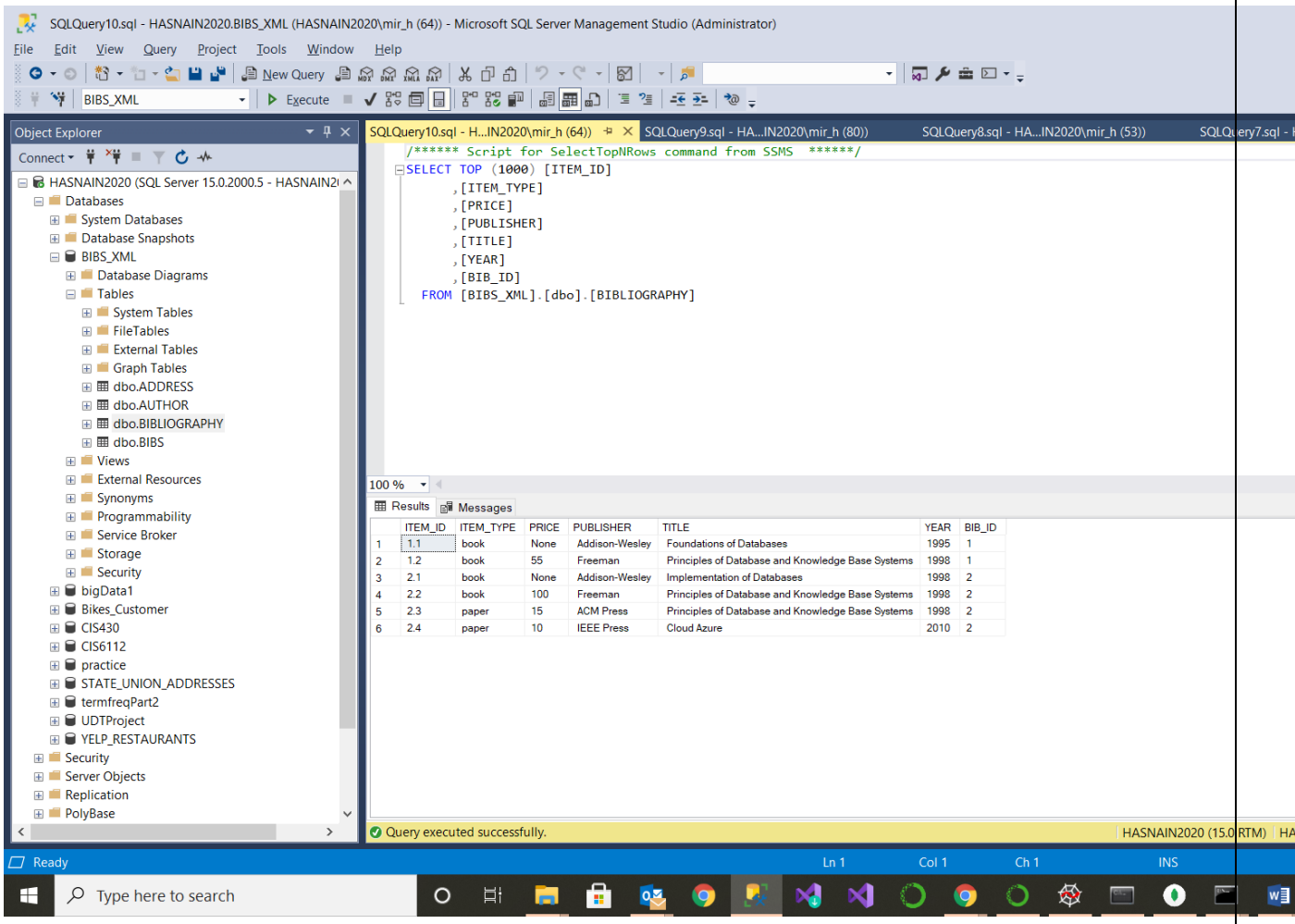

bibs.csv - Excel
bib_id
bib_name
1 bib
2 bib
bibs


SQLQuery11.sql - HASNAIN2020.BIBS_XML (HASNAIN2020\mir_h (55)) - Microsoft SQL Server Management Studio (Administrator)
BIBS_XML
/****** Script for SelectTopNRows command from SSMS ******/
SELECT TOP (1000) [BIB_ID]
,[BIB_NAME]
FROM [BIBS_XML].[dbo].[BIBS]
BIB_ID BIB_NAME
1 1 bib
2 2 bib
Query executed successfully.

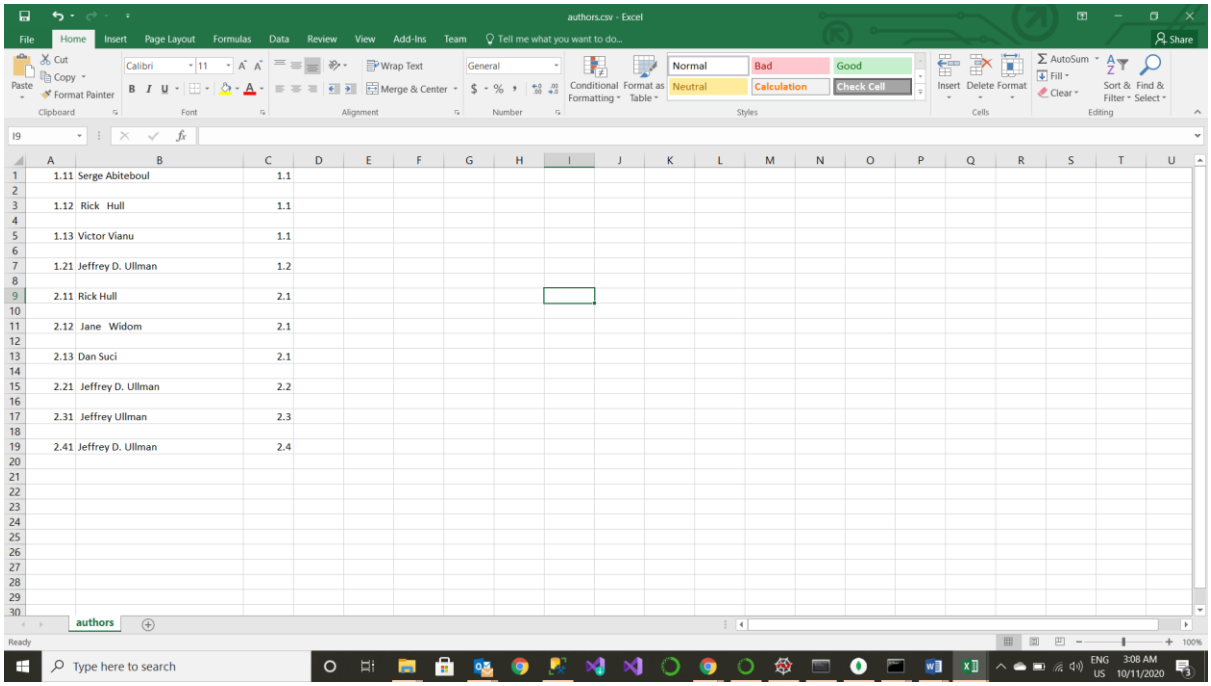
The bibliography.csv file consists of 6 entries into 7 . These are later put into a sql table.



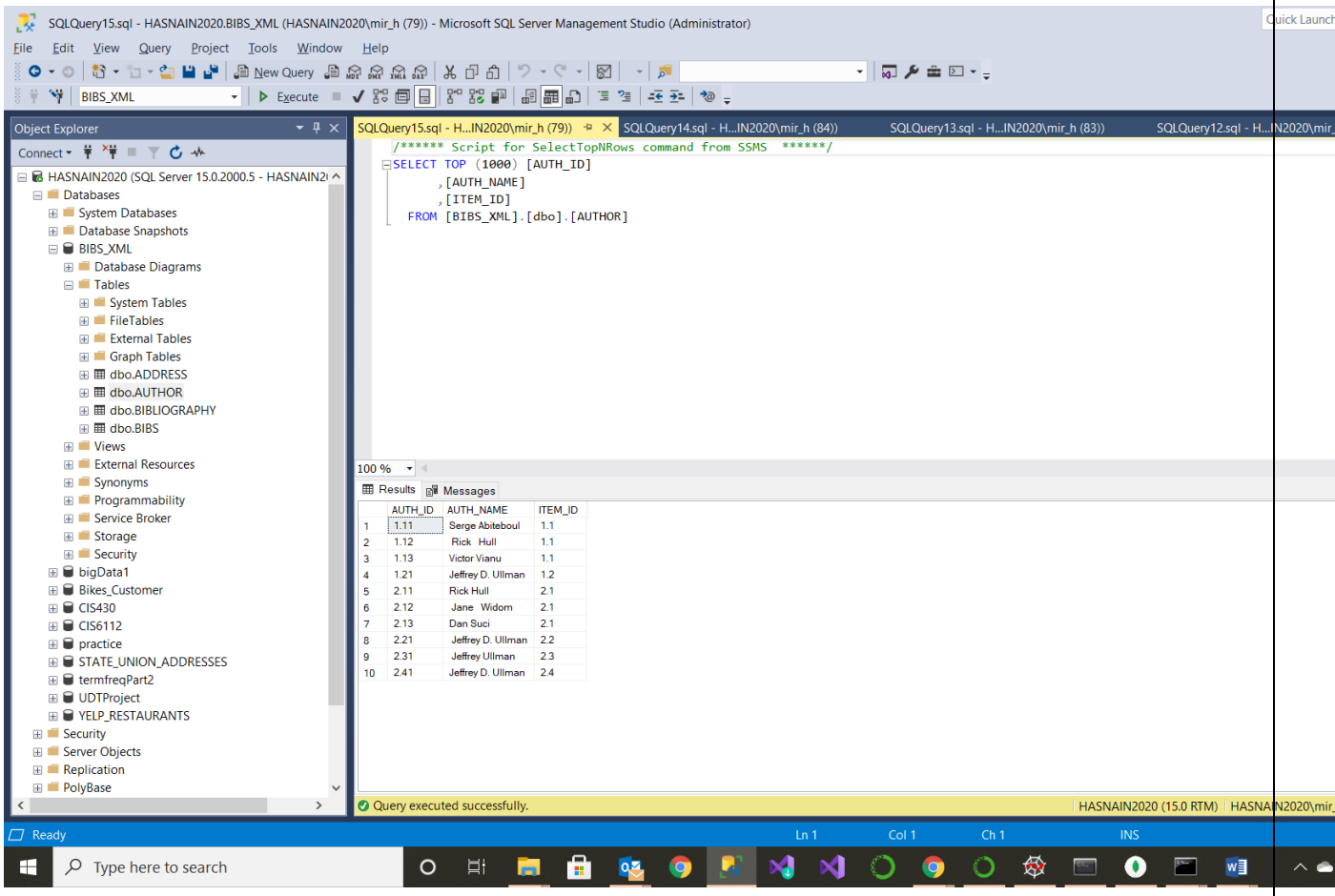
1.1	book		Addison-V	Foundatic	1995	1
1.2	book	55	Freeman	Principles	1998	1
2.1	book		Addison-V	Implemen	1998	2
2.2	book	100	Freeman	Principles	1998	2
2.3	paper	15	ACM Pres	Principles	1998	2
2.4	paper	10	IEEE Press	Cloud Azu	2010	2



The authors.csv file consists of 10 entries into 3 . These are later put into a sql table



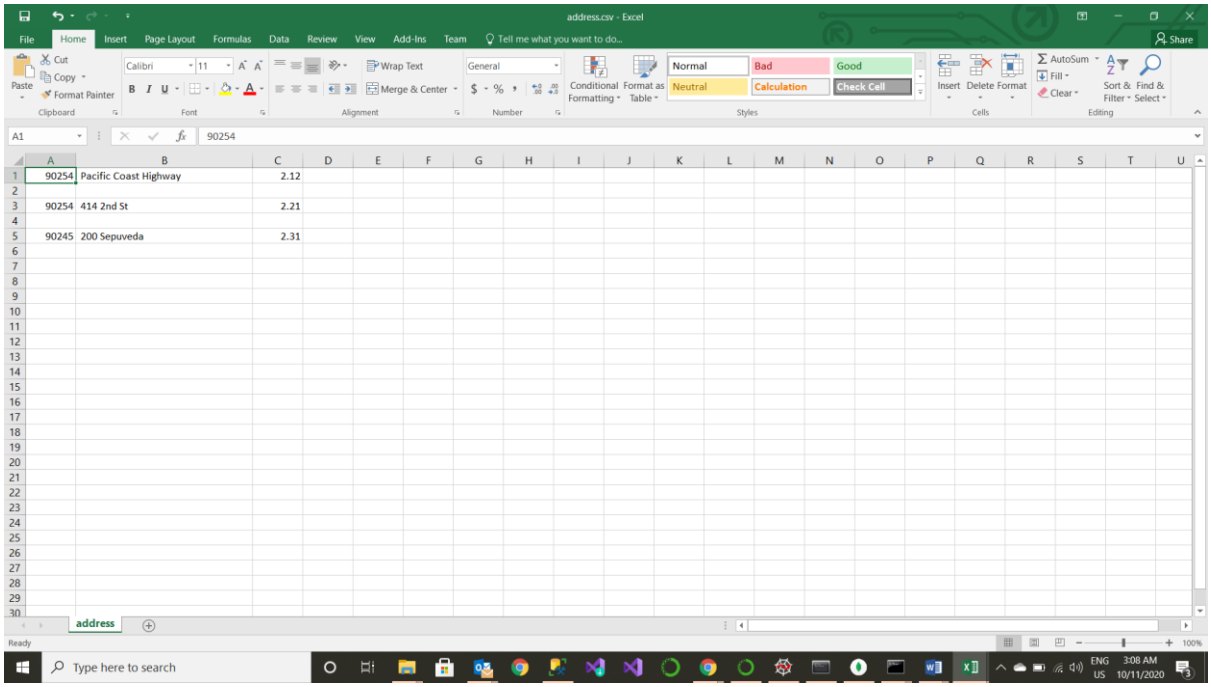
1.11	Serge Abiteboul	1.1
1.12	Rick Hull	1.1
1.13	Victor Vianu	1.1
1.21	Jeffrey D. Ullman	1.2
2.11	Rick Hull	2.1
2.12	Jane Widom	2.1
2.13	Dan Suci	2.1
2.21	Jeffrey D. Ullman	2.2
2.31	Jeffrey Ullman	2.3
2.41	Jeffrey D. Ullman	2.4

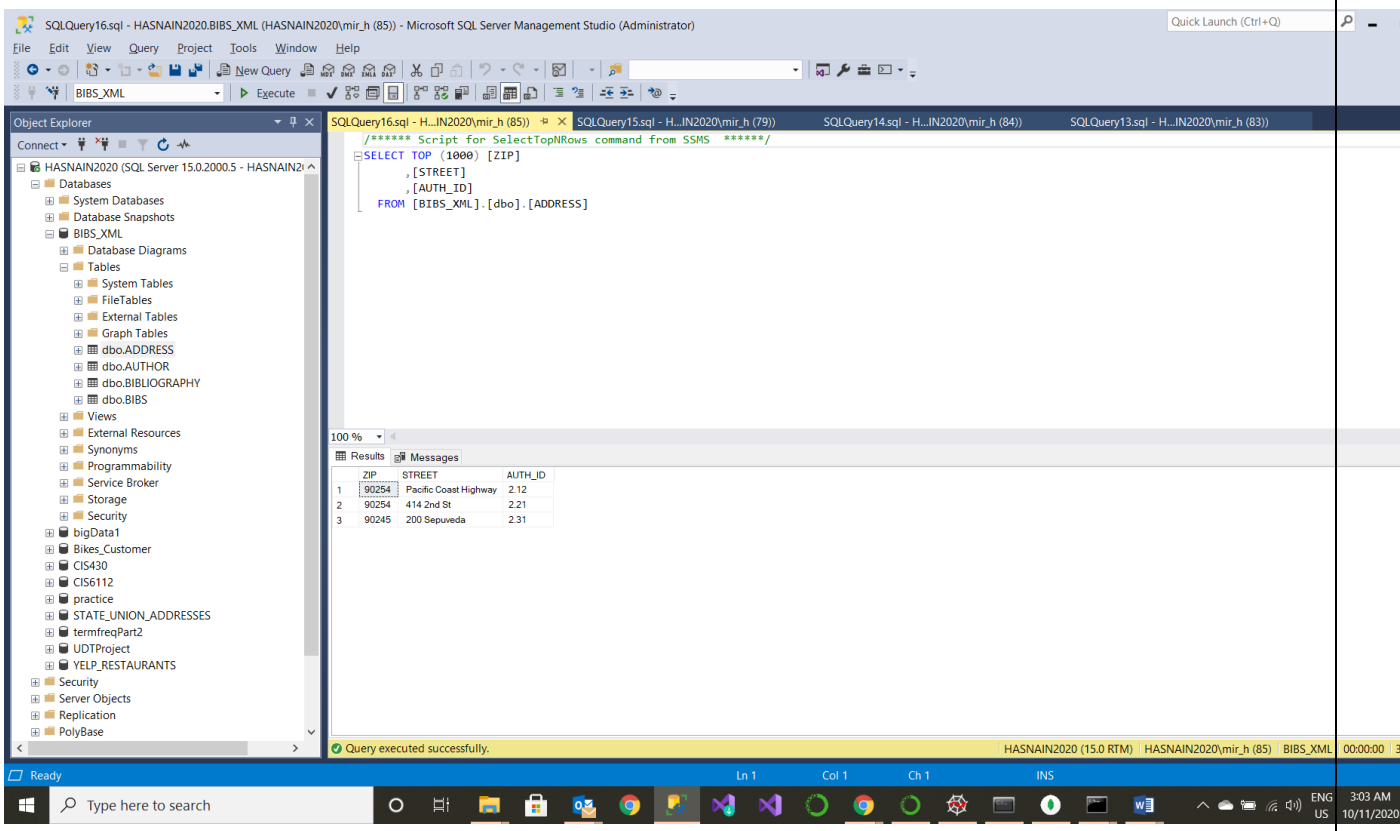


```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [AUTH_ID]
      ,[AUTH_NAME]
      ,[ITEM_ID]
  FROM [BIBS_XML].[dbo].[AUTHOR]
```

	AUTH_ID	AUTH_NAME	ITEM_ID
1	1.11	Serge Abiteboul	1.1
2	1.12	Rick Hull	1.1
3	1.13	Victor Vianu	1.1
4	1.21	Jeffrey D. Ullman	1.2
5	2.11	Rick Hull	2.1
6	2.12	Jane Widom	2.1
7	2.13	Dan Suci	2.1
8	2.21	Jeffrey D. Ullman	2.2
9	2.31	Jeffrey Ullman	2.3
10	2.41	Jeffrey D. Ullman	2.4

The address.csv file consists of 3 entries into 3 columns . These are later put into a sql table





Source Code:

```
import pandas as pd
import pyodbc
import xml.etree.ElementTree as ET

import csv

server='HASNAIN2020'
database='BIBS_XML'

def connect_to_sql_server():
   #connect to sql server
   print("Connecting to the server")
   odbc_conn=pyodbc.connect('DRIVER={SQL
SERVER};SERVER='+server+';Trusted_Connection=yes;')
   odbc_conn.autocommit=True
   cursor=odbc_conn.cursor()

   #create db if does not exist
   transaction="IF DB_ID('{0}') IS  NULL CREATE DATABASE
{0};".format(database)
   cursor.execute(transaction)
   if(cursor==True):
      print("created  db")

   transaction="USE {0}".format(database)
   cursor.execute(transaction)
   if(cursor==True):
      print("USe  db")

   #drop table if exists
   transaction="IF OBJECT_ID('dbo.ADDRESS') IS NOT  NULL DROP TABLE
dbo.ADDRESS;"
   cursor.execute(transaction)
   transaction="IF OBJECT_ID('dbo.AUTHOR') IS NOT  NULL DROP TABLE
dbo.AUTHOR;"
   cursor.execute(transaction)
```

```
    transaction="IF OBJECT_ID('dbo.BIBLIOGRAPHY') IS NOT  NULL DROP
TABLE dbo.BIBLIOGRAPHY;"
    cursor.execute(transaction)
    transaction="IF OBJECT_ID('dbo.BIBS') IS NOT  NULL DROP TABLE
dbo.BIBS;"
    cursor.execute(transaction)




    #create Bibs table
    transaction="IF OBJECT_ID('dbo.BIBS') IS NULL CREATE TABLE dbo.BIBS
(BIB_ID VARCHAR(10) PRIMARY KEY,BIB_NAME VARCHAR(15) );"
    cursor.execute(transaction)
    print("bibs table created")


    #create Bibliography table
    transaction="IF OBJECT_ID('dbo.BIBLIOGRAPHY') IS  NULL CREATE
TABLE dbo.BIBLIOGRAPHY (ITEM_ID VARCHAR(10) PRIMARY KEY,
ITEM_TYPE VARCHAR(10), PRICE VARCHAR(7), PUBLISHER VARCHAR(45),
TITLE VARCHAR(75), YEAR INT, BIB_ID VARCHAR(10) NOT
NULL,CONSTRAINT BIBLIOGRAPHY_fk_group FOREIGN KEY (BIB_ID)
REFERENCES dbo.BIBS(BIB_ID));"
    cursor.execute(transaction)
    print("Bibliography table created")

    #create author table
    transaction="IF OBJECT_ID('dbo.AUTHOR') IS  NULL CREATE TABLE
dbo.AUTHOR (AUTH_ID VARCHAR(20) PRIMARY KEY, AUTH_NAME
VARCHAR(60), ITEM_ID VARCHAR(10) NOT NULL, CONSTRAINT
AUTHOR_fk_group FOREIGN KEY (ITEM_ID) REFERENCES
dbo.BIBLIOGRAPHY(ITEM_ID));"
    cursor.execute(transaction)
    print("author table created")

    #create address table
```

```
    transaction="IF OBJECT_ID('dbo.ADDRESS') IS  NULL CREATE TABLE
dbo.ADDRESS (ZIP INT, STREET VARCHAR(60), AUTH_ID VARCHAR(20)
NOT NULL, CONSTRAINT ADDRESS_fk_group FOREIGN KEY (AUTH_ID)
REFERENCES dbo.AUTHOR(AUTH_ID));"
    cursor.execute(transaction)
    print("address table created")

    return cursor




tree = ET.parse('BibInputFile.xml')
root = tree.getroot()


bibs_df=pd.DataFrame(columns=["bib_id", "bib_name"])
bibliography_df=pd.DataFrame(columns=["item_id","item_type","price"
,"publisher","title","year", "bib_id"])
author_df=pd.DataFrame(columns=["auth_id","auth_name", "item_id"])
address_df=pd.DataFrame(columns=["zip", "street","auth_id"])

cursor=connect_to_sql_server()

for iteration,bib in enumerate(list(root)):

    bib_id=iteration+1
    print(bib.tag,bib_id)
    items=list(bib)
    #bibs_df=bibs_df.append([bib_id,bib.tag.strip()],ignore_index=True)

    #inserting into bibs table
    transaction="INSERT INTO dbo.BIBS VALUES(
'{0}','{1}');".format(bib_id,bib.tag)
    cursor.execute(transaction)

    #writing to csv file
```

```
    with open('bibs.csv','a') as bibsfile:
        writer=csv.DictWriter(bibsfile,["bib_id", "bib_name"])
        if iteration==0:
            writer.writeheader()
        writer.writerow({'bib_id':bib_id,'bib_name':bib.tag})

    for iteration2,item in enumerate(items):

        item_id=bib_id+(iteration2+1)/10
        print(item.tag,item_id)
        #print(list(item))
        item_type=item.tag
        price=item.attrib.get('price')
        publisher=item.find('publisher').text
        title=item.find('title').text
        year=item.find('year').text

#bibliography_df=bibliography_df.append([item_id,item_type,price,publ
isher,title,year,bib_id],ignore_index=True)
        print(item_id,item_type,price,publisher,title,year,bib_id)
        #inserting into bibiliography table
        transaction="INSERT INTO dbo.BIBLIOGRAPHY
VALUES('{0}','{1}','{2}','{3}','{4}','{5}','{6}');".format(item_id,item_type,pric
e,publisher,title,year,bib_id)
        cursor.execute(transaction)

        #writing to csv file
        with open('bibliography.csv','a') as itemsfile:
            writer=csv.writer(itemsfile)

writer.writerow([item_id,item_type,price,publisher,title,year,bib_id])

        #getting the authors
        authors=item.findall('author')
        for iter3, author in enumerate(authors):
            auth_id=item_id+(iter3+1)*0.01
            auth_id=round(auth_id,2)
```

```
        authNames=list(author)
        if(authNames==[]):
            auth_name=(author.text).strip()
        else:
            auth_name=[a.text for a in authNames if a.text]
            auth_name=(' ').join(auth_name)
        print(auth_id,auth_name,item_id)
        #inserting into authors table
        transaction="INSERT INTO dbo.AUTHOR
VALUES('{0}','{1}','{2}');".format(auth_id,auth_name,item_id)
        cursor.execute(transaction)

        #writing to csv file
        with open('authors.csv','a') as authorsfile:
            writer=csv.writer(authorsfile)
            writer.writerow([auth_id,auth_name,item_id])

         #getting the address of authors
        addresses=author.find('address')
        if addresses:
            zipcode=addresses.find('zip').text
            street=addresses.find('street').text
            print(zipcode,street,auth_id)
            #inserting into ADDRESS table
            transaction="INSERT INTO dbo.ADDRESS
VALUES('{0}','{1}','{2}');".format(zipcode,street,auth_id)
            cursor.execute(transaction)

            #writing to csv file
            with open('address.csv','a') as addressfile:
                writer=csv.writer(addressfile)
                writer.writerow([zipcode,street,auth_id])
```