Felipe Gomez

04/23/2024

BIDD 330A

Module 03

GitHub BIDD 330_Spring2024 Link: https://github.com/Phillips094/BIDD330_Spring2024

SQL Server Database Import/Export & Jupyter Notebook SQL Exercises Using Python

Introduction:

For module 03, we focus on developing a data warehouse from raw csv files into our localhost instance on SQL Server. The two files we use to import into our LocalHost instance for SQL Server are “bing_covid-19_data.csv” and “Unemployment.csv”. We utilize stored procedures to clean up our data and develop our data warehouse, and once this is developed, we focus on exporting our database (after dropping our stored procedures) into the UW Azure SQL Server database. Once our data warehouse has been imported successfully into our UW Azure SQL Server database, we create a Jupyter notebook to write Python code to run SQL scripts on our dimensions and fact tables.

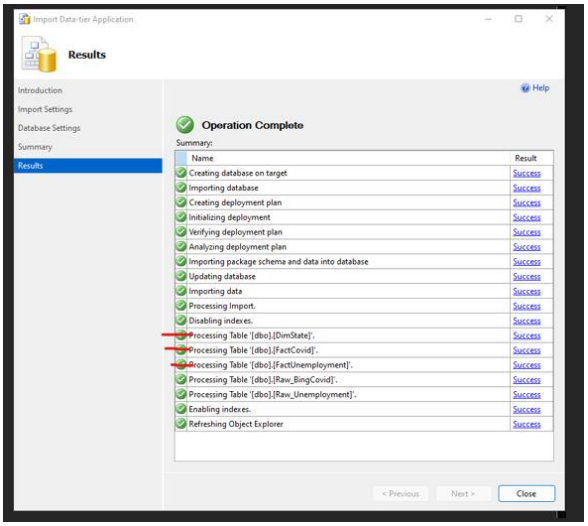
Starting off our module 03 project, we download our necessary files from Microsoft’s online data source and from our Canvas site. We first create a database that is tailored to our Final project group color and our name, i.e. my database name is “Gold_Felipe”. After creating our database we use the SQL Server Import and export Wizard to ingest our two csv files into our database. Note that we rename these two source tables by prefixing them with “Raw_” so we recognize that these are the raw files that we use to create our data warehouse. We save a backup of this database in our C drive and then move on to importing our new database into our UW server.

During this process, it is important to note that we use “Export Data-tier Application” and “Import Data-tier Application” as our tools for importing and exporting our database. When we import our database to the UW Server, we now have our ability to create our Dimension and Fact tables. Our final data warehouse contains DimDates, DimCountry, DimStates, FactCovid and FactUnemployment. We develop stored procedures to complete this. Before moving on to the next part of the project, we must also test our data warehouse to ensure that our stored procedures worked as intended. We must perform a data quality check to make sure our data warehouse is filled up with the correct data and correct number of rows.

An example of one of our stored procedures is as below:

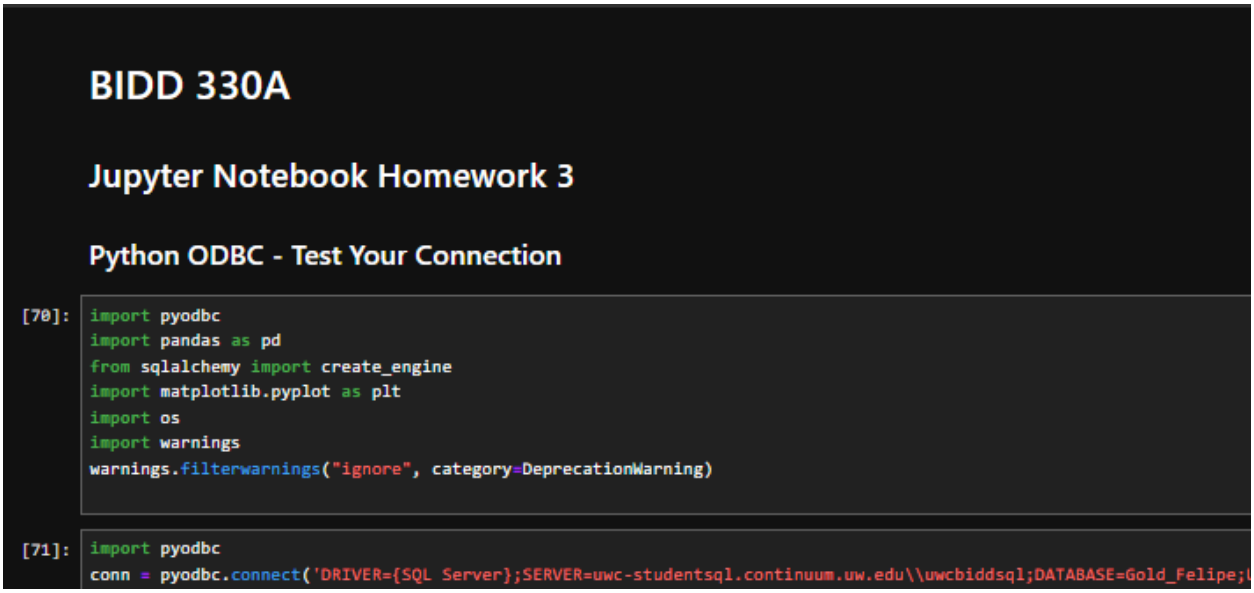
```sql
CREATE PROCEDURE [dbo].[sp_FactCovid]
AS
/*************
*Developer: Felipe Gomez
*Date: 04/16/2024
*Description: ETL Process for Raw Fact Covid
*
*
***************/

--Step 2: Drop Table
DROP TABLE IF EXISTS [dbo].[FactCovid]

--Step 3: Create Table with updated types
CREATE TABLE [dbo].[FactCovid]
(
      FactCovid_Key int IDENTITY (1,1) --future Surrogate Key
      ,ID INT
      ,Updated date
      ,Confirmed int
      ,Confirmed_Change int
      ,Deaths int
      ,Deaths_Change int
      ,Recovered int
      ,Recovered_Change int
      ,Latitude nvarchar(50)
      ,Longitude nvarchar(50)
      ,Iso2 nvarchar(50)
      ,Iso3  nvarchar(50)
      ,Country_Region nvarchar(50)
      ,Admin_Region_1 nvarchar(50)
      ,Iso_Subdivision nvarchar(50)
      ,Admin_Region_2 nvarchar(50)
      --,Load_Time date
)
--Step 4:Write ETL fixing datatypes
INSERT INTO [dbo].[FactCovid]
SELECT
--TOP 10  --Please start with TOP 10 rows. This will speed up your attempts
      CAST([id] as INT)
      , CAST([updated] as date)
      , CAST([confirmed] as INT)
      , CAST([confirmed_change] as INT)
      , CAST([deaths] as INT)
      , CAST([deaths_change] as INT)
      , CAST([recovered] as INT)
      , CAST([recovered_change] as INT)
      , [latitude]
      , [longitude]
      , [iso2]
      , [iso3]
      , [country_region]
      , [admin_region_1]
      , [iso_subdivision]
      , [admin_region_2]
      --, CAST([load_time] as date)
FROM [Black_Unemployment].[dbo].[Raw_BingCovid] RAW WITH (NOLOCK)
GO
```

After we have finished creating our data warehouse, we are now able to Drop our stored procedures so that we can perform another Export Data-tier Application for submitting this assignment.



We then move on to our second part of our assignment. We turn our attention to writing Python in a Jupyter notebook where we use python to write code to connect to our UW server. We use python to connect to our Gold_Felipe database so that we can begin querying our data warehouse and run

queries. We import our libraries that we need to utilize to write functions that will allow us to query our fact and dimension tables. We also must ensure we have our correct credentials and make sure that our conn variable in our 2nd cell runs successfully. After establishing this connection, we perform some simple Top 10 SELECT queries to make sure that our code is running smoothly and our connection works.



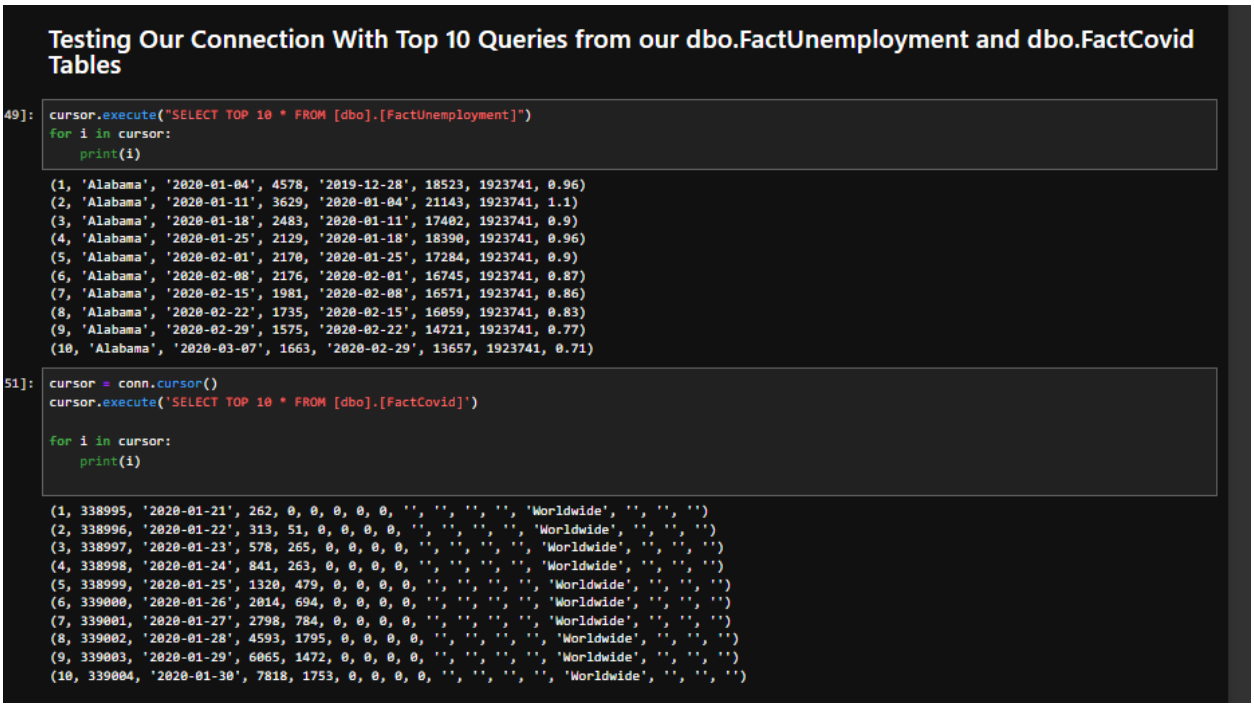
BIDD 330A

Jupyter Notebook Homework 3

Python ODBC - Test Your Connection

[70]:
```
import pyodbc
import pandas as pd
from sqlalchemy import create_engine
import matplotlib.pyplot as plt
import os
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

[71]:
```
import pyodbc
conn = pyodbc.connect('DRIVER={SQL Server};SERVER=uwc-studentsql.continuum.uw.edu\\uwcbiddsql;DATABASE=Gold_Felipe;
```



Testing Our Connection With Top 10 Queries from our dbo.FactUnemployment and dbo.FactCovid Tables

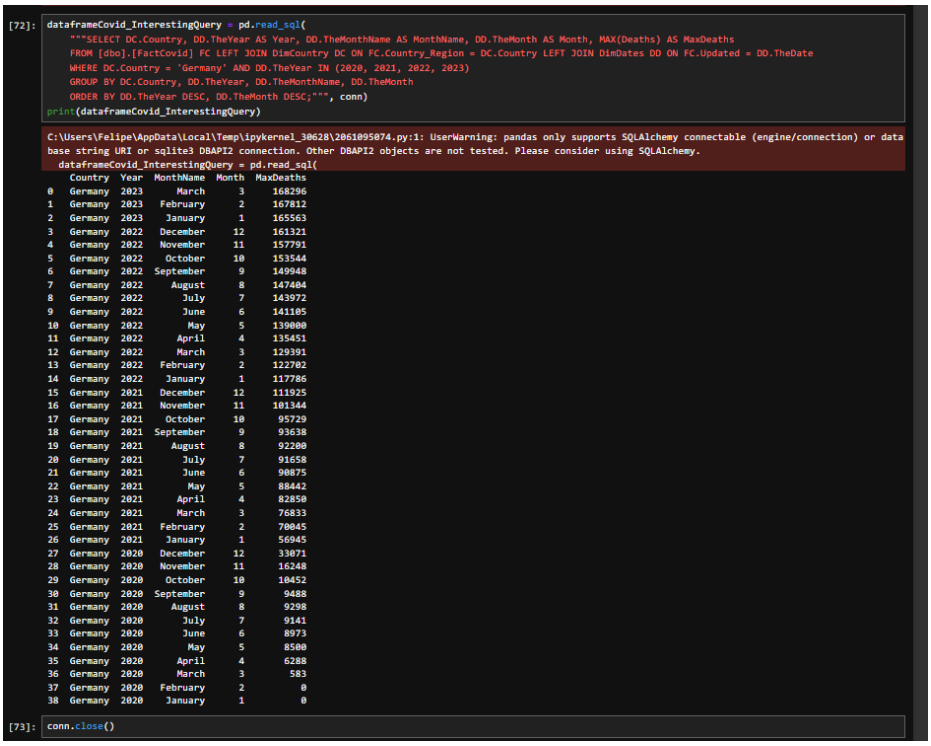
49]:
```
cursor.execute("SELECT TOP 10 * FROM [dbo].[FactUnemployment]")
for i in cursor:
    print(i)
```

```
(1, 'Alabama', '2020-01-04', 4578, '2019-12-28', 18523, 1923741, 0.96)
(2, 'Alabama', '2020-01-11', 3629, '2020-01-04', 21143, 1923741, 1.1)
(3, 'Alabama', '2020-01-18', 2483, '2020-01-11', 17402, 1923741, 0.9)
(4, 'Alabama', '2020-01-25', 2129, '2020-01-18', 18390, 1923741, 0.96)
(5, 'Alabama', '2020-02-01', 2170, '2020-01-25', 17284, 1923741, 0.9)
(6, 'Alabama', '2020-02-08', 2176, '2020-02-01', 16745, 1923741, 0.87)
(7, 'Alabama', '2020-02-15', 1981, '2020-02-08', 16571, 1923741, 0.86)
(8, 'Alabama', '2020-02-22', 1735, '2020-02-15', 16059, 1923741, 0.83)
(9, 'Alabama', '2020-02-29', 1575, '2020-02-22', 14721, 1923741, 0.77)
(10, 'Alabama', '2020-03-07', 1663, '2020-02-29', 13657, 1923741, 0.71)
```

51]:
```
cursor = conn.cursor()
cursor.execute('SELECT TOP 10 * FROM [dbo].[FactCovid]')

for i in cursor:
    print(i)
```

```
(1, 338995, '2020-01-21', 262, 0, 0, 0, 0, 0, '', '', '', '', 'Worldwide', '', '', '')
(2, 338996, '2020-01-22', 313, 51, 0, 0, 0, 0, '', '', '', '', 'Worldwide', '', '', '')
(3, 338997, '2020-01-23', 578, 265, 0, 0, 0, 0, '', '', '', '', 'Worldwide', '', '', '')
(4, 338998, '2020-01-24', 841, 263, 0, 0, 0, 0, '', '', '', '', 'Worldwide', '', '', '')
(5, 338999, '2020-01-25', 1320, 479, 0, 0, 0, 0, '', '', '', '', 'Worldwide', '', '', '')
(6, 339000, '2020-01-26', 2014, 694, 0, 0, 0, 0, '', '', '', '', 'Worldwide', '', '', '')
(7, 339001, '2020-01-27', 2798, 784, 0, 0, 0, 0, '', '', '', '', 'Worldwide', '', '', '')
(8, 339002, '2020-01-28', 4593, 1795, 0, 0, 0, 0, '', '', '', '', 'Worldwide', '', '', '')
(9, 339003, '2020-01-29', 6065, 1472, 0, 0, 0, 0, '', '', '', '', 'Worldwide', '', '', '')
(10, 339004, '2020-01-30', 7818, 1753, 0, 0, 0, 0, '', '', '', '', 'Worldwide', '', '', '')
```

Our last query that we test is meant to be an interesting query where we come up with a complex query that will be interesting to analyze. Here we decide to analyze the maximum number of accumulated covid deaths in Germany by month over time from 2020 to 2023. Essentially I want to see the accumulation of covid deaths on a monthly bases over time from descending order so we see the highest number of deaths in Germany.



```
[72]: dataframeCovid_InterestingQuery = pd.read_sql(
          """SELECT DC.Country, DD.TheYear AS Year, DD.TheMonthName AS MonthName, DD.TheMonth AS Month, MAX(Deaths) AS MaxDeaths
          FROM [dbo].[FactCovid] FC LEFT JOIN DimCountry DC ON FC.Country_Region = DC.Country LEFT JOIN DimDates DD ON FC.Updated = DD.TheDate
          WHERE DC.Country = 'Germany' AND DD.TheYear IN (2020, 2021, 2022, 2023)
          GROUP BY DC.Country, DD.TheYear, DD.TheMonthName, DD.TheMonth
          ORDER BY DD.TheYear DESC, DD.TheMonth DESC;""", conn)
      print(dataframeCovid_InterestingQuery)

C:\Users\Felipe\AppData\Local\Temp\ipykernel_30628\2061095074.py:1: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or data
base string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  dataframeCovid_InterestingQuery = pd.read_sql(
    Country  Year  MonthName  Month  MaxDeaths
0   Germany  2023      March      3     168296
1   Germany  2023   February      2     167812
2   Germany  2023    January      1     165563
3   Germany  2022   December     12     161321
4   Germany  2022   November     11     157791
5   Germany  2022    October     10     153544
6   Germany  2022  September      9     149948
7   Germany  2022     August      8     147404
8   Germany  2022       July      7     143972
9   Germany  2022       June      6     141105
10  Germany  2022        May      5     139000
11  Germany  2022      April      4     135451
12  Germany  2022      March      3     129391
13  Germany  2022   February      2     122702
14  Germany  2022    January      1     117786
15  Germany  2021   December     12     111925
16  Germany  2021   November     11     101344
17  Germany  2021    October     10      95729
18  Germany  2021  September      9      93638
19  Germany  2021     August      8      92200
20  Germany  2021       July      7      91658
21  Germany  2021       June      6      90875
22  Germany  2021        May      5      88442
23  Germany  2021      April      4      82850
24  Germany  2021      March      3      76833
25  Germany  2021   February      2      70045
26  Germany  2021    January      1      56945
27  Germany  2020   December     12      33071
28  Germany  2020   November     11      16248
29  Germany  2020    October     10      10452
30  Germany  2020  September      9       9488
31  Germany  2020     August      8       9298
32  Germany  2020       July      7       9141
33  Germany  2020       June      6       8973
34  Germany  2020        May      5       8500
35  Germany  2020      April      4       6288
36  Germany  2020      March      3        583
37  Germany  2020   February      2          0
38  Germany  2020    January      1          0

[73]: conn.close()
```

Summary:

In summary, we demonstrated that we can quickly develop a data warehouse from scratch by initially utilizing our LocalHost instance so we can then migrate over to our cloud server and begin developing our data warehouse. It seems that this method is very powerful and achieves quick, reliable results. I personally find this method of developing a data warehouse fascinating because we do not have to go through hoops and spend a lot of time trying to architect a solution from beginning to end that encompasses an ETL solution like SSIS or complex SQL code in our final product. In the end we are able to simply utilize SQL and Python code to analyze our data.