Extracción de datos de Acciones de una empresa con Python y R para su importación a una Base de Datos en SQL server.

Por: Cesar Arturo Ulloa Torres

“Crea, Automatiza, Utiliza y Transforma”

Parte 1: Ejecución del proceso en Python usando Pandas, pyodbc y sqlalchemy

Cesar Arturo
Ulloa Torres

Cesar Arturo Ulloa Torres

Primero, creamos el script que nos permitirá conectarnos a nuestra base de datos en SQL

In [ ]:
```
import pandas as pd #librería para gestionar dataframes
import pyodbc #librería que sirve para conectarse a diferentes tipos de bases de datos
from sqlalchemy import create_engine #librería encargada de crear el engine

# Configuración de la conexión (en este caso usa los parámetros del propio windows)
server = "LAPTOP-Arturo" #especificar el nombre del servidor con el que te vas a conectar
database = "Acciones" #el nombre de la base de datos con la que vas a trabajar

# Conectar la base de datos
conn_str = f"mssql+pyodbc://@{server}/{database}?driver=ODBC+Driver+17+for+SQL+Server" engine =
create_engine(conn_str)
```

Acabamos de generar una variable denominada "engine" que es nuestra "llave" para conectarnos con la base de datos SQL server. Cabe destacar que esta es una versión simplificada que no solicita ni usuario ni contraseña, usa los parámetros de inicio de sesión de Windows

Ahora, vamos paso a paso a subir un dataframe a SQL, primero debemos importar uno, usaremos los datos históricos de acciones de una empresa:

In [ ]:
```
#importamos la librería para acceder a los datos de Yahoo Finance
import yfinance as yf

Ticker = 'NVDA'  #este "Ticker" es el código con el que se identifica a la empresa
Nvidia1 = yf.download(Ticker, start='2020-01-01', end='2024-12-31')

Nvidia1.head(10) #nos mostrará las primeras filas
```

Out[ ]:

Price	Close	High	Low	Open	Volume
Ticker	NVDA	NVDA	NVDA	NVDA	NVDA
Date					
2020-01-02	5.972162	5.972162	5.892753	5.943286	237536000
2020-01-03	5.876571	5.920383	5.827531	5.852424	205384000
2020-01-06	5.901215	5.906442	5.757083	5.783220	262636000
2020-01-07	5.972659	6.018463	5.884537	5.929594	314856000
2020-01-08	5.983861	6.025184	5.928349	5.968427	277108000
2020-01-09	6.049580	6.122020	5.995811	6.070242	255112000
2020-01-10	6.081941	6.187240	6.067752	6.156870	316296000
2020-01-13	6.272624	6.297767	6.142432	6.165085	319840000
2020-01-14	6.155626	6.255199	6.142432	6.229808	359088000
2020-01-15	6.113058	6.190725	6.087169	6.168321	263104000

Vemos que las columnas que se generaron son de más de un nivel, esto podría generar problemas para

importarlo a SQL, antes de eso debemos eliminar el segundo nivel y asegurarnos que se importe correctamente

In [ ]:
```
# eliminamos el segundo nivel (el que tiene el nombre de las acciones)
Nvidia1.columns = Nvidia1.columns.droplevel(1)

#verificamos que se haya eliminado
Nvidia1.head(10)
```

Out[ ]:

Price	Close	High	Low	Open	Volume
Date					
2020-01-02	5.972162	5.972162	5.892753	5.943286	237536000
2020-01-03	5.876571	5.920383	5.827531	5.852424	205384000
2020-01-06	5.901215	5.906442	5.757083	5.783220	262636000
2020-01-07	5.972659	6.018463	5.884537	5.929594	314856000
2020-01-08	5.983861	6.025184	5.928349	5.968427	277108000
2020-01-09	6.049580	6.122020	5.995811	6.070242	255112000
2020-01-10	6.081941	6.187240	6.067752	6.156870	316296000
2020-01-13	6.272624	6.297767	6.142432	6.165085	319840000
2020-01-14	6.155626	6.255199	6.142432	6.229808	359088000
2020-01-15	6.113058	6.190725	6.087169	6.168321	263104000

In [18]:
```
Nvidia1.to_sql('Nvidia', engine, if_exists='replace', index=True)
```

Out[18]: 210

Esta última función lo que hace en términos generales es importar el dataframe llamado "Nvidia1" a una tabla en SQL server llamada "Nvidia", en caso esta tabla ya exista se reemplazará con la data de este nuevo dataframe, el parámetro de "index" está en "True" para que se importe la fecha, que en el dataframe la reconoce como el índice (la columna 0)

Ahora probemos si resultó, usemos el "engine" para realizar una consulta a la base de datos, tal como haríamos si estuvieramos en la interfaz de SSMS (SQL Server Management Studio)

In [33]:
```
Nvidia2023 = pd.read_sql("""
                         select * from Nvidia where
                         YEAR(Date)=2023
                         order by Date ASC""", engine)

# Mostrar las primeras filas
Nvidia2023.head(10)
```

Out[33]:

	Date	Close	High	Low	Open	Volume
0	2023-01-03	14.303279	14.983722	14.084458	14.838840	401277000
1	2023-01-04	14.736925	14.840840	14.229342	14.555075	431324000
2	2023-01-05	14.253321	14.552076	14.136416	14.479135	389168000
3	2023-01-06	14.846834	14.997711	14.022510	14.462149	405044000
4	2023-01-09	15.615206	16.042855	15.128604	15.271488	504231000
5	2023-01-10	15.895974	15.948930	15.459332	15.494303	384101000
6	2023-01-11	15.987899	16.014877	15.550258	15.827031	353285000
7	2023-01-12	16.497480	16.623377	15.479315	16.086818	551409000
8	2023-01-13	16.885164	16.908146	16.151766	16.264672	447287000
9	2023-01-17	17.687506	17.713485	16.885164	16.885164	511102000

La función read_sql que pertenece a Pandas, permite redactar una consulta como si se tratara de SQL server, esto nos permite extraer datos usando consultas desde las mas sencillas hasta las mas complejas, usando como "llave", el "engine" que habíamos usado antes.

Por último, con el objetivo de demostrar que efectivamente la consulta funcionó y se extrajo los datos de las acciones únicamente para el año 2023

In [ ]:
```
#extrae el primer dato que se identifica (fila número 0)
primer_fila = Nvidia2023.head(1)

#extrae el último dato del que se tiene registro (fila número 249)
ultima_fila = Nvidia2023.tail(1)

df_unido = pd.concat([primer_fila, ultima_fila], axis=0) df_unido
```

Out[ ]:

	Date	Close	High	Low	Open	Volume
0	2023-01-03	14.303279	14.983722	14.084458	14.838840	401277000
249	2023-12-29	49.503410	49.978234	48.732700	49.794301	389293000

Cesar Arturo Ulloa Torres

Parte 2: Ejecución del proceso en R usando DBI, odbc y quantmod

Cesar Arturo Ulloa Torres

Cesar Arturo Ulloa Torres

Para realizar un proceso similar en R, usamos los paquetes DBI para gestionar bases de datos y odbc para realizar la conexión por medio de controladores, siendo el equivalente a pyodbc en python

```
library(DBI)
library(odbc)

## Warning: package 'odbc' was built under R version 4.4.3

# Definir parámetros de conexión
server <- "LAPTOP-Arturo"    # nombre del servidor
database <- "Acciones"       # nombre de la base de datos

# Conectar a la base de datos usando ODBC
con <- dbConnect(odbc(),
                 Driver   = "ODBC Driver 17 for SQL Server",
                 Server   = server,
                 Database = database,
                 Trusted_Connection = "Yes")  # Usar autenticación de Windows
```

Ahora, usamos el paquete quantmod, que funciona de manera similar a yfinance, con esta extraemos los datos de acciones

```
library(quantmod)

# Definir el Ticker
Ticker <- "NVDA"

# Descargar los datos de Yahoo Finance
getSymbols(Symbols = Ticker,
            src = "yahoo",
            from = as.Date("2020-01-01"),
            to = as.Date("2024-12-31"),
            auto.assign = TRUE)

# Los datos se guardan en un objeto llamado igual que el ticker: NVDA
head(NVDA, 10)  # Mostrar las primeras 10 filas

##            NVDA.Open NVDA.High NVDA.Low NVDA.Close NVDA.Volume NVDA.Adjusted
## 2020-01-02   5.96875   5.99775  5.91800    5.99775   237536000      5.972160
## 2020-01-03   5.87750   5.94575  5.85250    5.90175   205384000      5.876571
## 2020-01-06   5.80800   5.93175  5.78175    5.92650   262636000      5.901216
## 2020-01-07   5.95500   6.04425  5.90975    5.99825   314856000      5.972658
## 2020-01-08   5.99400   6.05100  5.95375    6.00950   277108000      5.983861
## 2020-01-09   6.09625   6.14825  6.02150    6.07550   255112000      6.049580
## 2020-01-10   6.18325   6.21375  6.09375    6.10800   316296000      6.081941
## 2020-01-13   6.19150   6.32475  6.16875    6.29950   319840000      6.272625
## 2020-01-14   6.25650   6.28200  6.16875    6.18200   359088000      6.155626
## 2020-01-15   6.19475   6.21725  6.11325    6.13925   263104000      6.113057
```

En el caso de R, no existe como tal columnas multinivel, pero vemos que en cada columna hay un prefijo que indica el nombre de la empresa, para eliminarlo, convertir el tipo de tabla a un dataframe y reordenar las columnas del dataframe para que se asemejen al proceso realizado en python empleamos las siguientes funciones:

```
# Convertir de xts/zoo a data.frame clásico
Nvidia1 <- data.frame(Date = index(NVDA), coredata(NVDA))

# Eliminar el prefijo "NVDA." de los nombres de columnas
colnames(Nvidia1) <- gsub("NVDA\\.", "", colnames(Nvidia1))

# Reordenar las columnas manualmente
Nvidia1 <- Nvidia1[, c("Date", "Close", "High", "Low", "Open", "Volume")]

# Verificar el resultado
head(Nvidia1, 10)
```

Cesar Arturo Ulloa Torres

```
##          Date   Close    High     Low    Open    Volume
## 1  2020-01-02 5.99775 5.99775 5.91800 5.96875 237536000
## 2  2020-01-03 5.90175 5.94575 5.85250 5.87750 205384000
## 3  2020-01-06 5.92650 5.93175 5.78175 5.80800 262636000
## 4  2020-01-07 5.99825 6.04425 5.90975 5.95500 314856000
## 5  2020-01-08 6.00950 6.05100 5.95375 5.99400 277108000
## 6  2020-01-09 6.07550 6.14825 6.02150 6.09625 255112000
## 7  2020-01-10 6.10800 6.21375 6.09375 6.18325 316296000
## 8  2020-01-13 6.29950 6.32475 6.16875 6.19150 319840000
## 9  2020-01-14 6.18200 6.28200 6.16875 6.25650 359088000
## 10 2020-01-15 6.13925 6.21725 6.11325 6.19475 263104000
```

Ahora, con la tabla lista, procedemos a importar el dataframe a SQL usando los parámetros con los que hicimos la conexión inicialmente

```
library(DBI)
# Escribir el data.frame en la base de datos
dbWriteTable(con,
             name = "Nvidia",     # Nombre de la tabla en SQL Server
             value = Nvidia1,     # DataFrame que quieres guardar
             overwrite = TRUE,    # Reemplaza la tabla si ya existe
             row.names = FALSE)   # No guardar índices como columna
```

Ahora, realizamos la misma consulta de ejemplo para probar si se ha realizado la importación correctamente

```
library(DBI)
# Realizar la consulta SQL directamente
Nvidia2023 <- dbGetQuery(con, "
                         SELECT *
                         FROM Nvidia
                         WHERE YEAR(Date) = 2023
                         ORDER BY Date ASC
                         ")
# Verificar las primeras filas
head(Nvidia2023, 10)
```

```
##          Date  Close   High    Low   Open    Volume
## 1  2023-01-03 14.315 14.996 14.096 14.851 401277000
## 2  2023-01-04 14.749 14.853 14.241 14.567 431324000
## 3  2023-01-05 14.265 14.564 14.148 14.491 389168000
## 4  2023-01-06 14.859 15.010 14.034 14.474 405044000
## 5  2023-01-09 15.628 16.056 15.141 15.284 504231000
## 6  2023-01-10 15.909 15.962 15.472 15.507 384101000
## 7  2023-01-11 16.001 16.028 15.563 15.840 353285000
## 8  2023-01-12 16.511 16.637 15.492 16.100 551409000
## 9  2023-01-13 16.899 16.922 16.165 16.278 447287000
## 10 2023-01-17 17.702 17.728 16.899 16.899 511102000
```

Por último, para corroborar que la consulta fue generada correctamente, extraemos el primer y último valor del dataframe

```
primera_fila <- head(Nvidia2023, 1) # Extraer la primera fila

ultima_fila <- tail(Nvidia2023, 1) # Extraer la última fila

df_unido <- rbind(primera_fila, ultima_fila) # Unir ambas filas

print(df_unido) # Mostrar el resultado
```

```
##           Date  Close   High    Low   Open    Volume
## 1   2023-01-03 14.315 14.996 14.096 14.851 401277000
## 250 2023-12-29 49.522 49.997 48.751 49.813 389293000
```

Extracción de datos de Acciones de una empresa

arturo_ulloa0204

Cesar Arturo Ulloa Torres

arturoulloat@gmail.com