Documentacion de Implementación de API Yahoo Finance

Yahoo Finance API es una interfaz de programación de aplicaciones (API) que proporciona acceso a una amplia gama de datos financieros y de mercado. Esta API es ofrecida por Yahoo y es ampliamente utilizada en el mundo financiero y por desarrolladores para acceder a información en tiempo real y datos históricos relacionados con los mercados financieros. Aquí tienes una descripción de las principales características de la API de Yahoo Finance:

Datos en tiempo real: La API ofrece acceso a datos en tiempo real de una variedad de activos financieros, incluyendo acciones, índices, divisas y materias primas. Esto permite a los usuarios obtener información actualizada sobre precios, volumen y otros indicadores clave.

Datos históricos: Permite acceder a datos históricos de precios y volúmenes, lo que facilita el análisis y la visualización de tendencias a lo largo del tiempo.

Información detallada de acciones: La API proporciona información detallada sobre acciones individuales, incluyendo datos financieros, noticias relacionadas con la empresa, indicadores técnicos y más.

1. Extracción De Los Datos

Esto es una api la cual la podemos usar instalando e importando el modulo de yfinance. Con **“pip install yfinance”.**

Una vez instalado e importado con “**import yfinance”**, procedemos a extraer los datos, el **yf.ticker** llama a la función y retorna la acción o activo que le coloquemos en este caso **“ABNB”** que es Airbnb.

```
msft = yf.Ticker("ABNB")
df =  msft.history(period="MAX", interval="1d")
df
```

✓ 2.4s

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2020-12-10 00:00:00-05:00	146.000000	165.000000	141.250000	144.710007	70447500	0.0	0.0
2020-12-11 00:00:00-05:00	146.550003	151.500000	135.100006	139.250000	26980800	0.0	0.0
2020-12-14 00:00:00-05:00	135.000000	135.300003	125.160004	130.000000	16966100	0.0	0.0
2020-12-15 00:00:00-05:00	126.690002	127.599998	121.500000	124.800003	10914400	0.0	0.0
2020-12-16 00:00:00-05:00	125.830002	142.000000	124.910004	137.990005	20409600	0.0	0.0
...	...	...	...	...	...	...	...
2023-10-10 00:00:00-04:00	127.690002	132.779999	126.949997	131.589996	5114700	0.0	0.0
2023-10-11 00:00:00-04:00	132.119995	132.690002	128.520004	130.000000	3456500	0.0	0.0
2023-10-12 00:00:00-04:00	130.289993	130.410004	124.820000	125.970001	4023300	0.0	0.0
2023-10-13 00:00:00-04:00	127.610001	129.839996	123.550003	124.080002	4926500	0.0	0.0
2023-10-16 00:00:00-04:00	125.190002	126.760002	124.185997	125.190002	2704200	0.0	0.0

La Columna **"Date"** lo lee como índice porque en Pandas, es común utilizar una columna de fecha y hora como índice cuando se trabaja con datos de series de tiempo. Esto es especialmente útil cuando se hace realizar análisis y operaciones relacionadas con fechas, ya que tener la fecha y hora como índice permite un acceso más eficiente a los datos en función de la fecha.

por esta razón no lo colocamos como colomna, para el EDA, pero si cuando hagamos las transformaciones para subirlo a la base de datos.

Api_yf_transform.py

Funciones: A continuación, se describe cada función en el código y una breve explicación de su propósito y cómo se utiliza.

1. read_Data():

- **Descripción**: Esta función utiliza la biblioteca yfinance para recuperar los datos históricos de las acciones de Airbnb (ABNB).
- **Uso**: Llama a esta función para obtener un DataFrame con los datos históricos de ABNB.

2. cambiar_a_columna_date(df):

- **Descripción**: Esta función cambia el índice de un DataFrame a una columna y convierte los nombres de las columnas en minúsculas,esto lo hacemos con el fin de poder agregar la información correctamente a la base de datos.
- **Parámetros**: **df** es el DataFrame que se va a transformar.
- **Uso**: Llama a esta función con un DataFrame como argumento.

3. borrar_columns(df):

- **Descripción**: Esta función elimina las columnas "dividends" y "stock splits" del DataFrame, ya que como fue explicado en el EDA, estas columnas todos sus valores son 0, esto se puede deber a que esta compañía no reparte dividendos a sus accionistas y además en los ultimos 718 dias no han hecho splits a sus acciones.
- **Parámetros**: **df** es el DataFrame del que se eliminarán las columnas.
- **Uso**: Llama a esta función con un DataFrame que tenga las columnas que se desean eliminar.

load.py

Para ejecutar este código, asegúrese de tener Python instalado junto con las bibliotecas **psycopg2, SQLAlchemy** e importar las funciones definidas en el archivo **api_yf_transform**. Además, debe configurar una base de datos PostgreSQL con los parámetros de conexión adecuados y tener un contenedor de PostgreSQL en ejecución.

```
import psycopg2
from sqlalchemy import create_engine
from api_yf_transform import cambiar_a_columna_date, borrar_columns, read_Data
```

Funciones: A continuación, se describen las funciones y pasos clave del código:

1. call_to_transform ():

- **Descripción**: esta función llama a todas las funciones del archivo **api_yf_transform.py** que son: **read_Data(), cambiar_a_columna_date(df)** y **borrar_columns(df_previo) explicadas previamente .**
- **Uso**: Llama a esta función para realizar todo el proceso de transformación de los datos.

2. La Funcion load_data()

Intentar conectarse a la base de datos:

- **Descripción**: Se intenta establecer una conexión a la base de datos PostgreSQL utilizando los parámetros de conexión proporcionados.
- **Uso**: Esto se realiza automáticamente al ejecutar el código.

create_table_query:

- **Descripción**: Define la consulta SQL para crear una tabla en la base de datos PostgreSQL.
- **Uso**: Esta consulta se utiliza para crear una tabla llamada "apiyahoofinace" con columnas Date, Open, High, Low, Close y Volume.

Cargar los datos desde el DataFrame en la tabla:

- **Descripción**: Se utiliza la biblioteca SQLAlchemy para cargar el DataFrame final en la tabla de la base de datos PostgreSQL.
- **Uso**: se llama la funcion call_to_transform() y los datos resultantes quedan en el DataFrame **df_final** que son los que se cargan en la tabla "apiyahoofinace".

Consultar el catálogo information_schema para obtener las tablas:

- **Descripción**: Se ejecuta una consulta SQL para obtener una lista de las tablas en la base de datos PostgreSQL.
- **Uso**: Esto se utiliza para verificar si la tabla "apiyahoofinace" se creó correctamente.

Cerrar la conexión:

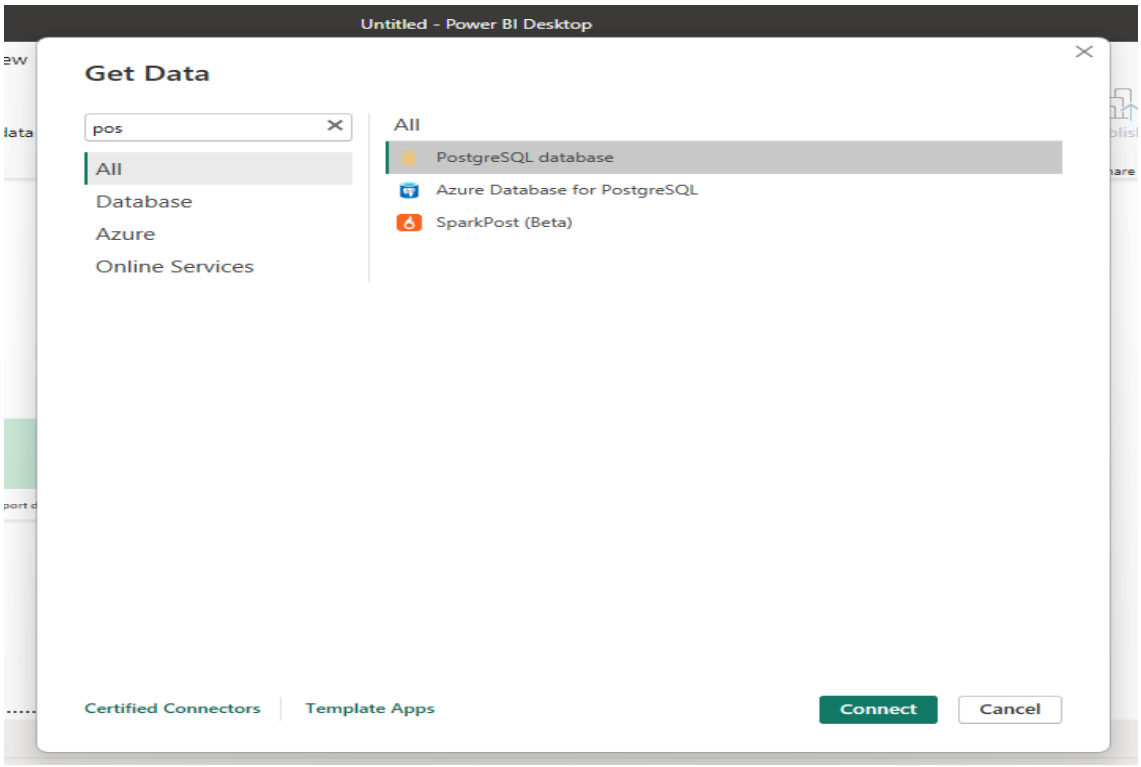
- **Descripción**: Se cierra la conexión a la base de datos PostgreSQL una vez que se completan todas las operaciones.
- **Uso**: Esto se realiza automáticamente al final del código.

7. El **if __name__ =="__main__": llama a la función** load_data().

Verificacmos que se crearon las Tablas y los datos fueron subidos a la base de Datos

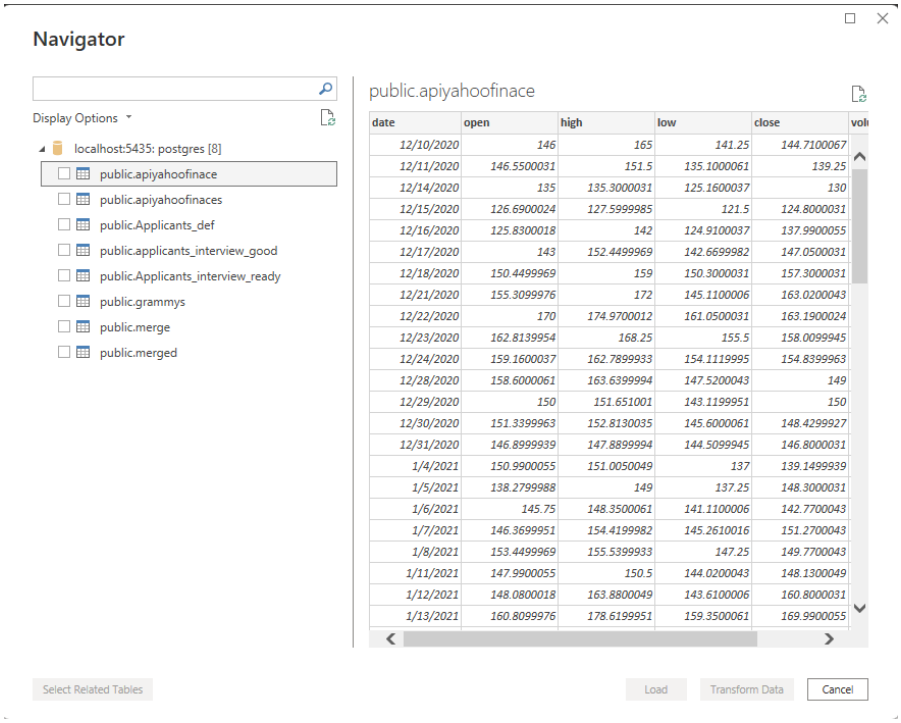
```
1   2020-12-11 00:00:00-05:00  146.550003  151.500000  135.100006  139.250000  26980800
2   2020-12-14 00:00:00-05:00  135.000000  135.300003  125.160004  130.000000  16966100
3   2020-12-15 00:00:00-05:00  126.690002  127.599998  121.500000  124.800003  10914400
4   2020-12-16 00:00:00-05:00  125.830002  142.000000  124.910004  137.990005  20409600
..                        ...         ...         ...         ...         ...       ...
711 2023-10-10 00:00:00-04:00  127.690002  132.779999  126.949997  131.589996   5114700
712 2023-10-11 00:00:00-04:00  132.119995  132.690002  128.520004  130.000000   3456500
713 2023-10-12 00:00:00-04:00  130.289993  130.410004  124.820000  125.970001   4023300
714 2023-10-13 00:00:00-04:00  127.610001  129.839996  123.550003  124.080002   4926500
715 2023-10-16 00:00:00-04:00  125.190002  126.760002  124.185997  125.190002   2704200

[716 rows x 6 columns]
Index(['date', 'open', 'high', 'low', 'close', 'volume'], dtype='object')
Tablas en la base de datos:
Applicants_def
Applicants_interview_ready
apiyahoofinace
merge
merged
apiyahoofinaces
applicants_interview_good
grammys
```

Conexión Para Graficar:



Graficando

Podemos observar de que efectivamente, se creo la tabla subieron los datos a la bd:



date	open	high	low	close	vol
12/10/2020	146	165	141.25	144.7100067	
12/11/2020	146.5500031	151.5	135.1000061	139.25	
12/14/2020	135	135.3000031	125.1600037	130	
12/15/2020	126.6900024	127.5999985	121.5	124.8000031	
12/16/2020	125.8300018	142	124.9100037	137.9900055	
12/17/2020	143	152.4499969	142.6699982	147.0500031	
12/18/2020	150.4499969	159	150.3000031	157.3000031	
12/21/2020	155.3099976	172	145.1100006	163.0200043	
12/22/2020	170	174.9700012	161.0500031	163.1900024	
12/23/2020	162.8139954	168.25	155.5	158.0099945	
12/24/2020	159.1600037	162.7899933	154.1119995	154.8399963	
12/28/2020	158.6000061	163.6399994	147.5200043	149	
12/29/2020	150	151.651001	143.1199951	150	
12/30/2020	151.3399963	152.8130035	145.6000061	148.4299927	
12/31/2020	146.8999939	147.8899994	144.5099945	146.8000031	
1/4/2021	150.9900055	151.0050049	137	139.1499939	
1/5/2021	138.2799988	149	137.25	148.3000031	
1/6/2021	145.75	148.3500061	141.1100006	142.7700043	
1/7/2021	146.3699951	154.4199982	145.2610016	151.2700043	
1/8/2021	153.4499969	155.5399933	147.25	149.7700043	
1/11/2021	147.9900055	150.5	144.0200043	148.1300049	
1/12/2021	148.0800018	163.8800049	143.6100006	160.8000031	
1/13/2021	160.8099976	178.6199951	159.3500061	169.9900055	