```python
import requests

# URL del archivo Parquet (ejemplo: datos de vuelos)
parquet_url = "https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2022-01.parquet"

# Descargar el archivo Parquet
response = requests.get(parquet_url)
with open("/tmp/flights.parquet", "wb") as f:
    f.write(response.content)

# Leer el archivo Parquet descargado
flights_df = spark.read.parquet("/tmp/flights.parquet")

# Mostrar el esquema y los primeros registros
flights_df.printSchema()
flights_df.show(5)

# Escritura de Parquetx
flights_df.write.parquet("hdfs://localhost:9000/user/hdfs/salida_parquet", mode="overwrite")
```

Este código es muy similar al ejemplo de lectura de CSV, pero en este caso, está diseñado para trabajar con archivos en formato Parquet, que es un formato de almacenamiento columnar eficiente para trabajar con grandes conjuntos de datos, especialmente en entornos como Spark. Vamos a analizarlo:

1. **`import requests`**: Importa la librería `requests` para realizar la descarga del archivo desde la URL.
2. **`from pyspark.sql import SparkSession`**: Importa la clase `SparkSession` para interactuar con Spark DataFrames.
3. **`spark = SparkSession.builder \`**: Inicializa o obtiene una `SparkSession`.
    - **`.appName("Parquet File Reading") \`**: Establece un nombre descriptivo para la aplicación Spark.
    - **`.getOrCreate()`**: Asegura que se utilice una `SparkSession` existente o se cree una nueva si no hay una activa.
4. **`parquet_url = "https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2022-01.parquet"`**: Define la URL del archivo Parquet que contiene datos de viajes (yellow taxi trip data de Nueva York para enero de 2022).
5. **`response = requests.get(parquet_url)`**: Realiza una petición HTTP GET a la URL para descargar el contenido del archivo Parquet.
6. **`with open("/tmp/flights.parquet", "wb") as f:`**: Abre un archivo local llamado `flights.parquet` en modo de escritura binaria (`"wb"`) en el directorio `/tmp/`. El uso de `with` garantiza que el archivo se cierre correctamente después de la operación.

7. **`f.write(response.content)`**: Escribe el contenido descargado del archivo Parquet en el archivo local.
8. **`flights_df = spark.read.parquet("/tmp/flights.parquet")`**: Utiliza el lector de Parquet de Spark (`spark.read.parquet()`) para cargar el archivo Parquet descargado en un DataFrame llamado `flights_df`. A diferencia de los CSV, Parquet es auto-esquematizante, por lo que generalmente no necesitas especificar el esquema explícitamente (aunque puedes hacerlo si es necesario). Spark infiere el esquema directamente del archivo Parquet.
9. **`flights_df.printSchema()`**: Muestra el esquema del DataFrame `flights_df`, incluyendo los nombres de las columnas y sus tipos de datos, tal como se definen en el archivo Parquet.
10. **`flights_df.show(5)`**: Muestra las primeras 5 filas del DataFrame para dar una idea del contenido de los datos.
11. **`flights_df.write.parquet("hdfs://localhost:9000/user/hdfs/salida_parquet", mode="overwrite")`**: Intenta escribir el DataFrame `flights_df` en formato Parquet en HDFS.
    - **`"hdfs://localhost:9000/user/hdfs/salida_parquet"`**: Especifica la ruta en HDFS donde se guardarán los archivos Parquet. Al igual que en el ejemplo de CSV, asume que un NameNode de HDFS está corriendo en `localhost:9000` y que tienes permisos para escribir en el directorio `/user/hdfs/salida_parquet`.
    - **`mode="overwrite"`**: Indica que si el directorio de destino ya existe, su contenido se eliminará antes de escribir los nuevos archivos.

Puntos Clave y Consideraciones (en España, siendo las 7:23 PM CET del 24 de marzo de 2025):

- **Dependencia de `requests`:** Asegúrate de tener instalada la librería `requests` (`pip install requests`).
- **Acceso a la URL:** Verifica que tu conexión a internet permita acceder a la URL del archivo Parquet.
- **Permisos en `/tmp/`:** El script necesita permisos para escribir el archivo temporal en el directorio `/tmp/` de tu sistema local.
- **Configuración de HDFS:** La escritura a HDFS solo funcionará si tienes un clúster Hadoop configurado y en ejecución en la dirección especificada (`hdfs://localhost:9000`).
    - Si no tienes un clúster Hadoop local, esta parte del código fallará con errores de conexión o relacionados con el sistema de archivos.
    - Comprueba que el NameNode esté activo y que el usuario que ejecuta el script tenga los permisos necesarios para escribir en la ruta de destino en HDFS.

Si tu objetivo principal es leer y procesar el archivo Parquet localmente sin interactuar con HDFS, puedes comentar o eliminar la última línea de código (`flights_df.write.parquet(...)`).

El formato Parquet es muy eficiente para el almacenamiento y procesamiento de datos en Spark debido a su naturaleza columnar, que permite una mejor compresión y un acceso más rápido a columnas específicas.