```python
pip install mysql-connector-python
```

Nos conectaremos al servidor MySQL y crearemos una nueva base de datos para nuestras pruebas.

```python
from pyspark.sql import SparkSession
import mysql.connector

# Crear una sesión de Spark con soporte para JDBC
spark = SparkSession.builder \
    .appName("MySQL Integration") \
    .getOrCreate()

# Parámetros de conexión
mysql_host = "localhost"
mysql_port = 3306
mysql_user = "hive"
mysql_password = "hive_password"

# Conectar a MySQL y crear una nueva base de datos
connection = mysql.connector.connect(
    host=mysql_host,
    port=mysql_port,
    user=mysql_user,
    password=mysql_password
)
cursor = connection.cursor()

# Crear una nueva base de datos
new_database = "pyspark_test_db"
cursor.execute(f"CREATE DATABASE IF NOT EXISTS {new_database}")
print(f"Base de datos '{new_database}' creada correctamente.")

# Cerrar la conexión
cursor.close()
connection.close()

# Configurar la URL JDBC para la nueva base de datos
jdbc_url = f"jdbc:mysql://{mysql_host}:{mysql_port}/{new_database}"
connection_properties = {
    "user": mysql_user,
    "password": mysql_password,
    "driver": "com.mysql.cj.jdbc.Driver"
}
```

```
print(f"Conexión JDBC configurada para la base de datos: {new_database}")
```

Este código realiza dos acciones principales:

1. **Instala el conector de MySQL para Python:** La línea `pip install mysql-connector-python` (aunque no se ejecuta directamente aquí) es una instrucción para instalar la librería necesaria para que Python se comunique con un servidor MySQL.
2. **Crea una base de datos MySQL:** El script de Python utiliza la librería `mysql.connector` para establecer una conexión con un servidor MySQL y ejecutar un comando SQL para crear una nueva base de datos llamada `pyspark_test_db` si aún no existe.
3. **Configura la conexión JDBC para Spark:** Finalmente, el código configura una URL JDBC y las propiedades de conexión necesarias para que Spark pueda interactuar con la base de datos MySQL recién creada.

Análisis Detallado:

- **`pip install mysql-connector-python`**: Como se mencionó, este comando instalaría el driver de MySQL para Python. Necesitas ejecutar esto en tu terminal *antes* de ejecutar el script de Python si aún no tienes la librería instalada.
- **`from pyspark.sql import SparkSession`**: Importa la clase `SparkSession` de PySpark.
- **`import mysql.connector`**: Importa la librería `mysql.connector`, que permite a Python interactuar con servidores MySQL.
- **`spark = SparkSession.builder ... .getOrCreate()`**: Crea o obtiene una sesión de Spark. **Es importante notar que para que Spark pueda conectarse a MySQL a través de JDBC, necesitas incluir el driver JDBC de MySQL en el classpath de Spark.** Esto se hace añadiendo la siguiente configuración al builder de la `SparkSession`:

  Python

  ```
  .config("spark.jars.packages", "mysql:mysql-connector-java:8.0.28")
  ```

  Esta línea le dice a Spark que descargue el paquete `mysql:mysql-connector-java:8.0.28` (el driver JDBC de MySQL) y lo incluya en su classpath. Asegúrate de usar la versión del driver que sea compatible con tu servidor MySQL.

- **Parámetros de Conexión:** Se definen las variables para la dirección del servidor MySQL (`mysql_host`), el puerto (`mysql_port`), el usuario (`mysql_user`) y la contraseña (`mysql_password`). **Asegúrate de que estos valores sean correctos para tu entorno de MySQL.** En este caso, se utilizan las credenciales típicas de "hive", lo cual podría ser relevante si tu MySQL está integrado con un entorno Hadoop/Hive.
- **Conexión a MySQL y Creación de la Base de Datos:**
  - `mysql.connector.connect(...)`: Establece una conexión con el servidor MySQL utilizando los parámetros definidos.
  - `connection.cursor()`: Crea un objeto cursor que permite ejecutar comandos SQL.
  - `cursor.execute(f"CREATE DATABASE IF NOT EXISTS {new_database}")`: Ejecuta un comando SQL para crear una nueva base de datos llamada `pyspark_test_db` si aún no existe.

    - `print(...)`: Informa que la base de datos se ha creado correctamente.
    - `cursor.close()` y `connection.close()`: Cierran el cursor y la conexión con el servidor MySQL, liberando recursos.
- **Configuración de la URL JDBC:**
    - `jdbc_url = f"jdbc:mysql://{mysql_host}:{mysql_port}/{new_database}"`: Construye la URL JDBC necesaria para que Spark se conecte a la base de datos `pyspark_test_db`.
    - `connection_properties = {...}`: Define un diccionario con las propiedades de conexión necesarias para JDBC, incluyendo el usuario, la contraseña y el nombre de la clase del driver JDBC de MySQL (`com.mysql.cj.jdbc.Driver`).
- **`print(...)`**: Muestra la URL JDBC configurada.

Para que este código funcione correctamente y Spark pueda interactuar con la base de datos MySQL, asegúrate de lo siguiente:

1. **Has instalado `mysql-connector-python`:** Ejecuta `pip install mysql-connector-python` en tu terminal.
2. **Tienes un servidor MySQL en funcionamiento y accesible** en la dirección y puerto especificados (`localhost:3306` por defecto).
3. **El usuario MySQL especificado (`hive`) existe y tiene los permisos necesarios** para crear bases de datos y realizar otras operaciones en el servidor MySQL. La contraseña (`hive_password`) también debe ser correcta para este usuario.
4. **Has incluido la configuración para descargar el driver JDBC de MySQL en tu `SparkSession` builder.**

Con estos pasos, tu entorno de PySpark estará configurado para interactuar con la base de datos MySQL que has creado. En los siguientes pasos, podrías crear tablas en esta base de datos e interactuar con ellas usando Spark a través de JDBC.